Automatische Prüfkörperauswertung in der digitalen Mammographie

Diplom – Abschlussarbeit
im Institut für Photoingenieurwesen und Medientechnik
an der Fachhochschule Köln

Autor
Andreas Schreiber
aus Köln
Mat.-Nr.: 11020509

Referent: Prof. Dr. Dietmar Kunz
Korreferent: Prof. Dr. Christian Blendl

Köln, im August 2005

Automatic analysis of a phantom in digital mammography

Thesis
at the Institute of
Imaging Sciences and Media Technology
University of Applied Sciences Cologne

Author
Andreas Schreiber
Cologne
Mat.-Number: 11020509

First Reviewer: Prof. Dr. Dietmar Kunz
Second Reviewer: Prof. Dr. Christian Blendl

Cologne, August 2005

Kurzbeschreibung:

Titel:	Automatische Prüfkörperauswertung in der digitalen Mammographie
Autor:	Andreas Schreiber
Referenten:	Prof. Dr. Dietmar Kunz / Prof. Dr. Christian Blendl
Zusammenfassung:	Ziel dieser Arbeit ist die Automatisierung von Prüfverfahren, welche in der PAS 1054 definiert sind und in der digitalen Mammographie ihre Anwendung finden. Dazu standen Aufnahmen eines Prototypen des in der PAS beschriebenen Prüfkörpers zur Verfügung, die mit dem Sectra MDM angefertigt wurden. Anhand dessen konnte mit der Programmiersprache Java und unter Verwendung der Software ImageJ ein Programm geschrieben werden, welches die automatische Analyse der Bilddaten ermöglicht. Eine übersichtliche Benutzeroberfläche und die Ausgabe von Ergebnissen, die auf das Wesentliche reduziert sind, gewährleisten eine zeitsparende und effektive Handhabung.
Stichwörter:	Detektion, digitale Mammographie, Konstanzprüfung, Prüfkörper
Sperrvermerk:	Die vorgelegte Arbeit unterliegt keinem Sperrvermerk.
Datum:	19.08.2005

Abstract:

Title:	Automatic analysis of a phantom in digital mammograhy
Author:	Andreas Schreiber
Reviewers:	Prof. Dr. Dietmar Kunz / Prof. Dr. Christian Blendl
Abstract:	The aim of this thesis is the automation of test procedures, which are defined in the PAS 1054 and which are applied in digital mammography. Therefore were images of a prototype of the phantom provided, which is described in the PAS. These images were taken with the Sectra MDM. On this basis it was possible to write a program in Java and with utilisation of ImageJ, which enables the automatic analysis of the images. The clearly arranged desktop and the output of results, which are reduced to the essential, ensure an effective workflow.
Keywords:	detection, digital mammography, constancy test, phantom
Remark of closure:	This presented thesis is not subject of a blocking period.
Date:	19.08.2005

Mein Dank richtet sich an

Herrn Prof. Dr. Dietmar Kunz
und Herrn Prof. Dr. Christian Blendl
für die freundliche Unterstützung während der Diplomarbeit

Herrn Mats Danielsson (PhD), SECTRA GmbH
Herrn Dr. Andreas Keizers, SECTRA GmbH

meine Familie und meine Freunde
für die Unterstützung während meines gesamten Studiums

1 Einleitung

Im Gegensatz zu anderen Bereichen der medizinischen Diagnostik hat der Einsatz digitaler Geräte in der Mammographie relativ lange auf sich warten lassen, was auf die hohen Anforderungen an die Bildqualität zurückzuführen ist. Seit einigen Jahren werden jedoch digitale Systeme entwickelt, die sich zunehmend gegen analoge Geräte durchsetzen. Dabei kommen je nach Hersteller unterschiedliche Technologien zur Bildaufnahme zum Einsatz. Möglichkeiten, die jedes dieser Geräte bietet, ist die digitale Bildbearbeitung und -speicherung, sowie das Versenden der Bilddaten.
Besonders im Hinblick auf den Einsatz digitaler Röntgengeräte in Screeningprogrammen, hat die regelmäßige Überprüfung von Strahlendosis und Bildqualität auf der Grundlage einschlägiger Normen einen hohen Stellenwert. [1, Seite 3]

1.2 Aufgabenstellung

Eine Norm, welche die Anforderungen und die Prüfverfahren für digitale Röntgeneinrichtungen in der Mammographie festlegt, ist die PAS 1054 (Publicly Availably Specification). Sie repräsentiert den gegenwärtigen Stand der technischen Qualitätssicherung in der digitalen Mammographie in Deutschland. Die Prüfungen bestehen zum größten Teil aus Bildbearbeitungs und –bildverarbeitungs-prozessen, denen eine Reihe von Berechnungen zugrunde liegen. Ziel dieser Arbeit ist es, Teile der Konstanzprüfung, wie sie in der PAS beschrieben ist, zu automatisieren, indem mit Hilfe einer Software die notwendigen Informationen aus den entsprechenden Bildern gewonnen und mit diesen die nachfolgenden Berechnungen durchgeführt werden. Dazu müssen mit dem zu untersuchenden Gerät Aufnahmen eines Prüfkörpers angefertigt werden, dessen Beschaffenheit in der PAS festgelegt ist. [2]

2 Material und Methoden

2.1 Sectra MDM

Der schematische Aufbau einer digitalen Mammographie-Einrichtung ist in Abbildung 1 dargestellt. Der Unterschied zur konventionellen Mammographie besteht im Einsatz eines elektronischen Detektors anstelle der Film-Folien-Kombination. Der Detektor absorbiert die einfallende Röntgenstrahlung und wandelt diese in elektrische Ladung um. Die elektrische Ladung wird anschließend in ein digitales Signal umgewandelt, welches die Bildinformation repräsentiert. [4]
Die Aufnahmen des Prüfkörpers, welche in dieser Diplomarbeit verwendet worden sind, wurden mit dem Sectra MDM (Micro Dose Mammography) System angefertigt. Das im Klinikum Krefeld befindliche Gerät basiert auf einer Scannertechnik, wobei die einzelnen Elemente des Detektors eine Pixelgröße von 50µm aufweisen. Das Bildformat, in dem die Bilddaten ausgegeben werden, ist das DICOM (Digital Imaging and Communications in Medicine) Format. [1, Seite 5]

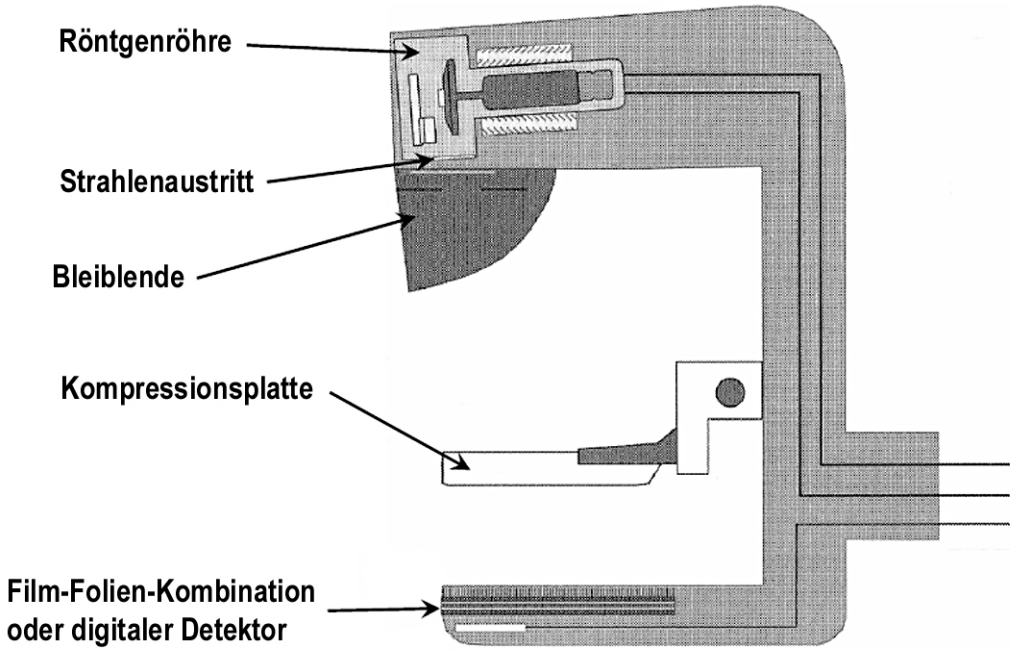

Abbildung 1: Schematischer Aufbau einer Mammographie-Einrichtung (modifiziert nach [3, Seite 7])

2.2 Der DICOM-Standard

Der DICOM-Standard ist zum Datenaustausch zwischen medizinisch-radiologischen Informationssystemen konzipiert. Es handelt sich um einen herstellerunabhängigen Standard, der es ermöglicht, digitale Bilder und die dazugehörigen Daten auf unterschiedlichen Systemen zu nutzen. Des weiteren gewährleistet er den Datenaustausch zwischen verschiedenen Informationssystemen, wie HIS (Hospital Information System) und RIS (Radiology Information System). "Er wurde nach dem OSI (Open System Interconnection) Modell entworfen, welches die Kommunikation zwischen heterogenen Systemen erlaubt. Um den fehlerfreien Austausch der Daten zu gewährleisten, enthält der DICOM Standard spezielle Protokolle, welche die Syntax und Semantik von Kommandos und Nachrichten beschreiben. Außerdem muss für jedes DICOM-kompatible Gerät eine exakte Beschreibung der Systemfähigkeit vorhanden sein [5]".
Die DICOM-Bilder können eine Bittiefe von maximal 16bit, was 65.536 Graustufen entspricht, darstellen. Jedoch wird meistens mit einer Bittiefe von 14bit gearbeitet, womit 16.384 Graustufen abgebildet werden können. Da die Bilddaten in unkomprimierter Form vorliegen und die Auflösung meist sehr hoch ist (z.B. Sectra MDM: 4740 x 5348 Pixel) sind die Datenmengen im Vergleich zu komprimierten Bilddatentypen recht hoch. Dies ist jedoch unumgänglich, da eine Komprimierung der Bilddaten zu Artefakten im Bild und somit zu einer Fehldiagnose führen könnten. Die hohe Auflösung spielt gerade bei der digitalen Mammographie eine wichtige Rolle, da es sich hier um sehr feine Strukturen handelt, die abgebildet und diagnostiziert werden müssen.
Der DICOM-Header, eine im Anhang des Bildes gespeicherte Zeichenfolge, liefert Informationen zum Bild, wie z.B. Patientendaten, Aufnahmeeinstellungen, Gerätetyp und Hersteller. Diese Informationen kann man bei Bedarf abrufen oder sie werden von einem DICOM-Viewer, einer Software zur Darstellung der DICOM-Bilder, direkt mit angezeigt.

2.3 Der Prüfkörper

Der Prüfkörper lässt sich aufteilen in einen Grundprüfkörper und eine Strukturplatte, die bei Bedarf auf diesen Grundprüfkörper aufgesetzt werden kann. Der Grundprüfkörper aus PMMA (Polymethylmetacrylat) hat eine Höhe von 40mm und die in Abbildung 2 dargestellten Abmessungen. Das in dieser Abbildung zu sehende Rechteck ist eine Aussparung für eine PMMA-Treppe, die bei Bedarf dort eingesetzt wird. Genauere Angaben zu dieser PMMA-Treppe und zu den an der Thoraxwandseite des Körpers befindlichen Stahlkugeln finden sich in den nachfolgenden Kapiteln. [2, Seite 44ff]

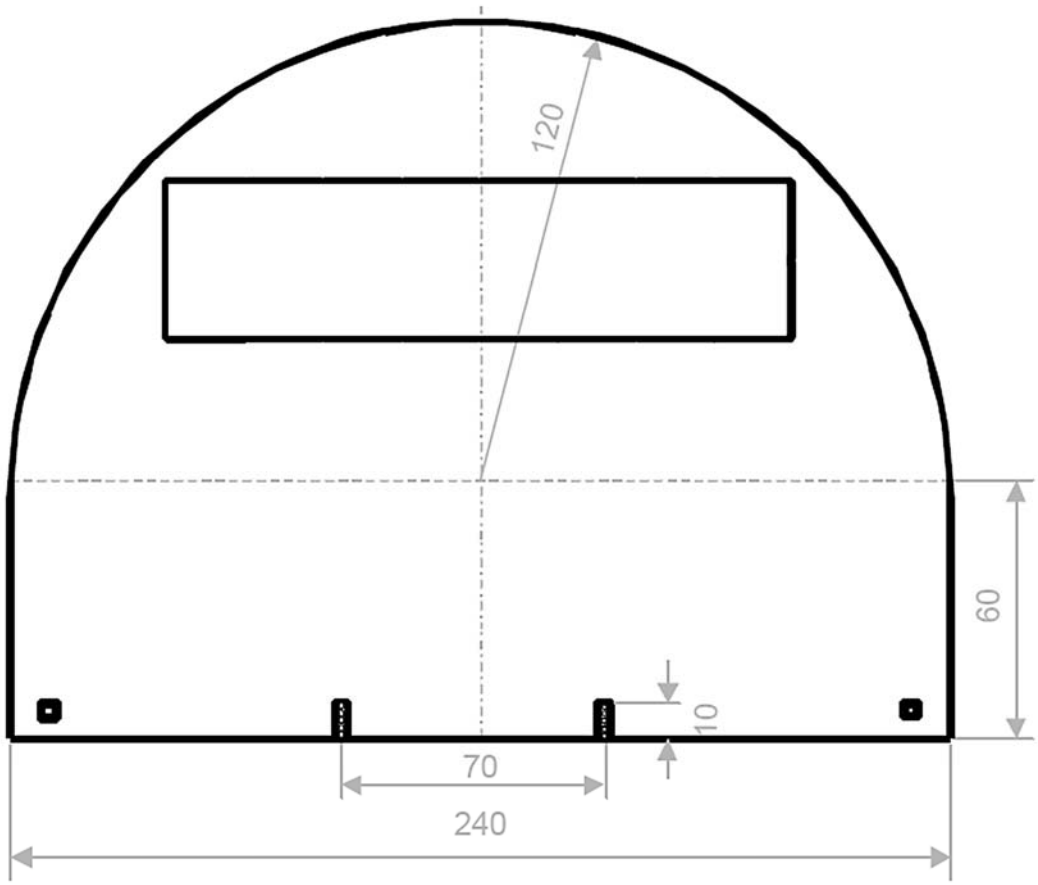

Abbildung 2: Maße des Grundprüfkörpers (modifiziert nach [2, Seite 49])

Die Strukturplatte, welche mit Hilfe von Passtiften genau auf dem Grundprüfkörper positioniert werden kann, ist ebenfalls aus PMMA gefertigt und hat eine Höhe von 6mm. Eine Aussparung mit einer Größe von ungefähr 80mm x 80mm ermöglicht das Einlegen unterschiedlicher Testeinsätze, die in den nachfolgenden Kapiteln genauer beschrieben werden. Weiterhin befindet sich auf der Strukturplatte ein 20mm x 20mm großes Quadrat, welches einen Bereich zur Messung des mittleren Grauwertes markiert. Zum Einsetzen eines Bleistrichrasters existiert eine runde Aussparung, die hier nicht weiter beschrieben wird, da sie für die Durchführung der in dieser Diplomarbeit automatisierten Prüfverfahren nicht von Bedeutung ist. Auch in der Strukturplatte befinden sich an der Thoraxwandseite zwei Stahlkugelreihen. [2, Seite 44]

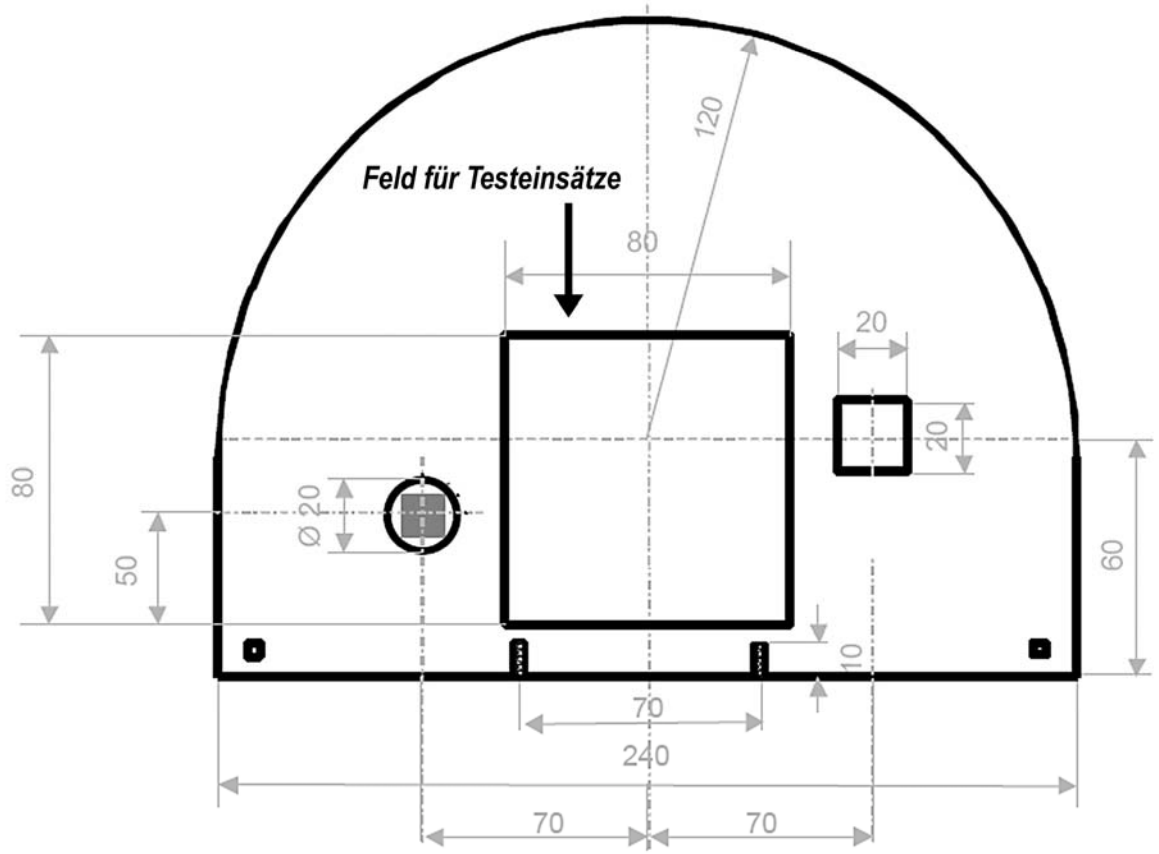

Abbildung 3: Maße der Strukturplatte (modifiziert nach[2, Seite 47])

2.4 Kurzbeschreibung der Prüfverfahren

In der PAS werden Anforderungen und Prüfverfahren beschrieben, mit denen die Bildqualität und die dazu benötigte Dosis in der digitalen Mammographie ermittelt werden können. Diese Verfahren werden sowohl zur Abnahme- als auch zur Konstanzprüfung solcher Anlagen verwendet. Die Häufigkeit mit der die einzelnen Prüfverfahren angewendet werden sollen, ist der PAS selber zu entnehmen. Bei einigen Prüfverfahren genügt eine jährliche Prüfung, andere dagegen müssen täglich, wöchentlich, monatlich oder mit der Häufigkeit, die der Hersteller vorsieht, durchgeführt werden. [2, Seite 50ff]
Das Durchführen solcher Prüfungen sichert die Bildqualität, reduziert die Strahlenbelastung des Patienten und verhindert unnötige Mehrfachbelichtungen.
Die in der PAS angegebenen Prüfverfahren, die in den Programmen behandelt wurden, sollen in den nachfolgenden Kapiteln kurz beschrieben werden. Alle weiteren Prüfverfahren oder genauere Anforderungen können in der PAS direkt nachgelesen werden.

2.4.1 Prüfung der thoraxwandseitigen Begrenzung

Die Prüfung der thoraxwandseitigen Begrenzung wird anhand einer Aufnahme des Grundprüfkörpers mit aufgesetzter Strukturplatte durchgeführt. Hierzu sind in dem Prüfkörper an der Thoraxwandseite die oben erwähnten Reihen mit jeweils fünf Stahlkugeln eingearbeitet, wobei die äußere Kugel bündig mit dem Prüfkörper abschließt. Jede Kugel hat eine Durchmesser von ca. 2mm. Zur sicheren Positionierung besitzt der Prüfkörper zwei Anschläge an dieser Seite. [2, Seite 44]
Für jede Stahlkugelreihe ist die Anzahl der ganz und halb abgebildeten Kugeln zu ermitteln. die Breite des nicht abgebildeten Bereiches darf höchstens 4mm betragen. Werden nicht genügend Kugeln vollständig abgebildet, so ist dies ein Hinweis darauf, dass der nicht abgebildete Bereich zu groß ist. [2, Seite 25] [6, Seite 9]

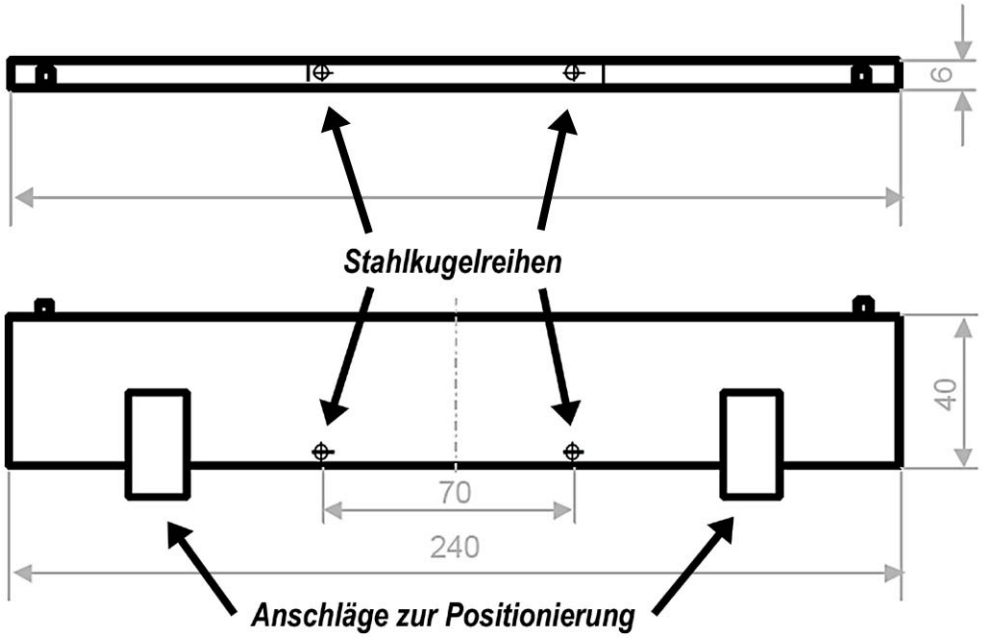

Abbildung 4: Querschnitt des Grundprüfkörpers und der Strukturplatte mit Lage der Stahlkugeln (modifiziert nach [2, Seite 46,47])

2.4.2 Bestimmung von SRV und KRV

Das SRV (Signal-Rausch-Verhältnis) ist definiert als das Verhältnis der vorhandenen mittleren Signalleistung zur vorhanden mittleren Rauschleistung, wobei der Ursprung der Rauschleistung nicht berücksichtigt wird:

$$srv = \frac{Nutzsignalleistung}{Rauschleistung} \tag{2.1}$$

Da die Signalleistung bei vielen technischen Anwendungen um mehrere Größenordnungen größer ist, als die Rauschleistung, wird das Signal-Rausch-Verhältnis häufig im logarithmischen Maßstab dargestellt: [9, Seite 134]

$$srv = 10 * \lg\left(\frac{Nutzsignalleistung}{Rauschleistung}\right) \tag{2.2}$$

Im Bezug auf die gegebene Prüfkörperaufnahme und die in der PAS angegebenen auszuwertenden Messgrößen ergibt sich hier für das SRV: [2, Seite 29]

$$srv = 20 * \lg\left(\frac{mittl.Grauwert}{Standardabweichung}\right) \tag{2.3}$$

Um das SRV und das KRV (Kontrast-Rausch-Verhältnis) zu bestimmen, wird eine Strukturplatte auf den Grundprüfkörper aufgebracht. In diese Strukturplatte wird für die Aufnahme zur Bestimmung des KRV der entsprechende Testeinsatz eingelegt. Dieser Einsatz besteht aus einer 6mm dicken PMMA-Platte die mit 0,1 mm Aluminium überlagert ist. Es befindet sich eine quadratische Markierung der Größe 20mm x 20mm 60mm von der Brustwandseite entfernt auf dem Testeinsatz, innerhalb welcher der mittlere Grauwert zu messen ist. Der KRV-Testeinsatz wird mittig zur Brustwandkante platziert. In dem auf dem Grundprüfkörper befindlichem 20mm x 20mm großen dafür vorgesehenen Bereich wird ebenfalls der mittlere Grauwert und die Standardabweichung gemessen. Aus diesen Werten und den Gleichungen 2.3 und 2.4 wird nun das SRV und KRV berechnet.
"Das KRV ist definiert als das Verhältnis der Differenz des mittleren Pixelwertes zweier unterschiedlich schwächender Objektbereiche (Kontrast) zu der Standardabweichung des mittleren Grauwertes eines der beiden Felder [2, Seite 9]". In Bezug auf die gegebene Prüfkörperaufnahme und die in der PAS angegebenen auszuwertenden Messgrößen ergibt sich für das KRV: [2, Seite 30]

$$krv = 20 * \lg\left(\frac{\left|mittl.Grauwert_{AL} - mittl.Grauwert_{PMMA}\right|}{Standardabweichung_{PMMA}}\right) \tag{2.4}$$

mittl. Grauwert$_{Al:}$	mittlerer Grauwert der 20mm x 20mm großen Fläche auf dem KRV-Testeinsatz
mittl. Grauwert$_{PMMA:}$	mittlerer Grauwert der 20mm x 20mm großen Fläche auf dem Grundprüfkörper
Standardabweichung$_{PMMA:}$	Standardabweichung der 20mm x 20mm großen Fläche auf dem Grundprüfkörper

"Neben der Prüfkörperdicke von 46mm PMMA müssen das SRV und das KRV von zwei weiteren Prüfkörperaufnahmen bestimmt werden, dessen Dicke etwa gleichabständig zur Bezugsdicke ist aber mindestens einen Unterschied von 14mm zu dieser haben muss [2, Seite 30]". Das KRV dieser beiden Aufnahmen darf nicht mehr als 10% vom Wert des am Grundprüfkörper bestimmten KRV abweichen. [2, Seite 29]

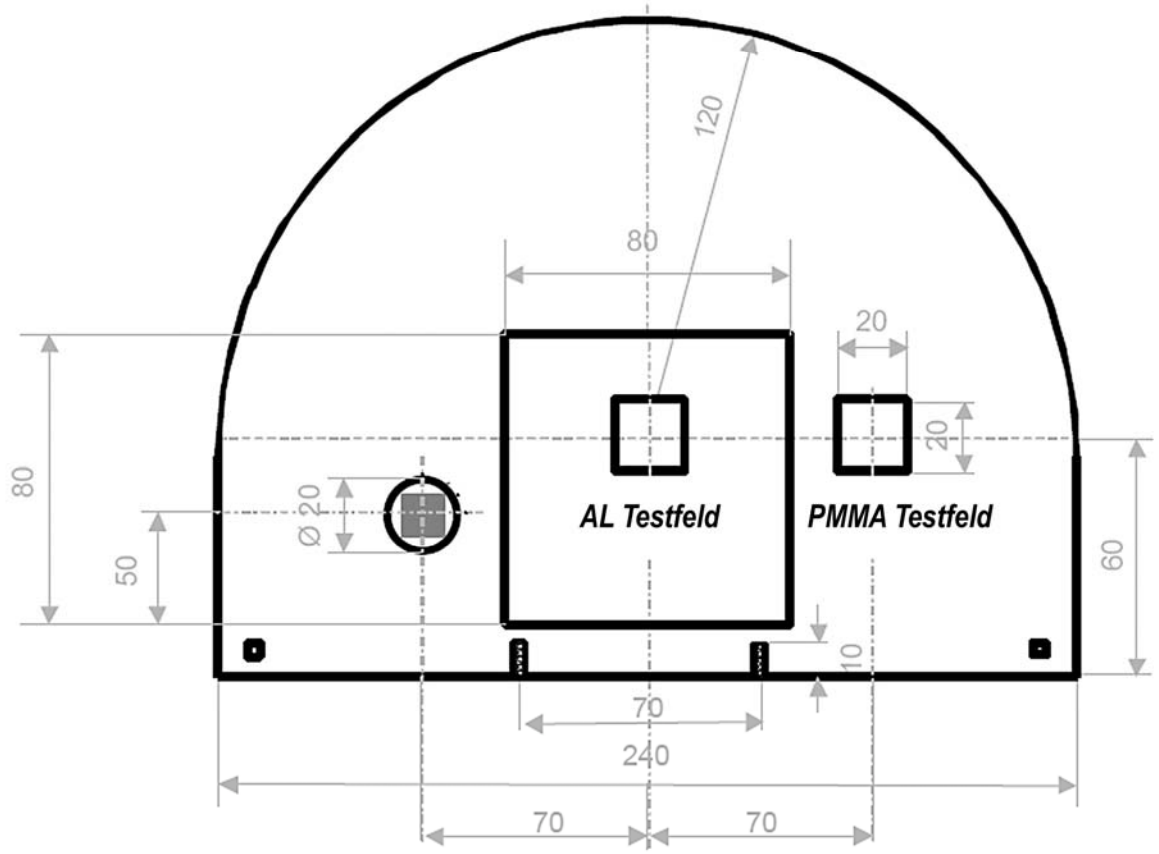

Abbildung 5: Strukturplatte mit Testeinsatz KRV (modifiziert nach [2, Seite 47])

2.4.3 Prüfung der Gleichförmigkeit

Zur Prüfung der Gleichförmigkeit wird eine Aufnahme einer 40mm dicken PMMA-Platte benötigt. Die Messungen sind an einem unkorrigierten Rohbild durchzuführen (umgangssprachlich: Roh-Roh-Bild). Es wird in sechs verschiedenen Bereichen über jeweils eine ROI (Region of Interest) der mittlere Grauwert bestimmt. Hierbei sollen drei ROI's an der brustwandseitigen Kante liegen und die anderen drei an der brustwandfernen Seite. Die Zentren der ROI's sollen sich 40mm weit entfernt von der jeweiligen Kante befinden. Der Abstand der äußeren vier ROI's zu den Seitenkanten soll ebenfalls 40mm betragen.
Von allen sechs Grauwerten wird der Mittelwert bestimmt. Die Grauwerte der einzelnen ROI's darf nicht mehr als 20% vom Mittelwert der sechs Grauwerte abweichen. [2, Seite 32ff]

2.4.4 Prüfung des Dynamikumfangs

Es wird eine Aufnahme des Grundprüfkörpers mit einer eingebetteten Treppe angefertigt. Es handelt sich hierbei um eine 14stufige PMMA-Treppe, deren Stufen jeweils eine Größe von 20mm x 20mm aufweisen. Die einzelnen Stufen weisen einen Unterschied von 3mm in ihrer Dicke auf, wobei die Stufe 0 einem Feld entspricht, bei welchem die Primärstrahlung ungeschwächt auf das Bildempfangssystem trifft. Zusätzlich ist neben der PMMA-Treppe eine 20mm x 40mm große Fläche eingearbeitet. Diese ist zur Bestimmung des Offsets mit einer totalabsorbierenden Folie, bestehend aus Blei, bedeckt. Von jedem der vierzehn Felder wird der mittlere Grauwert und die Standardabweichung bestimmt. An der Bleistufe, welche sich links neben der Treppe befindet, muss ebenfalls der Grauwert bestimmt werden. Diese Werte müssen nun mit Werten, die bei der Abnahmeprüfung des Gerätes ermittelt wurden, verglichen werden. Das Grauwertniveau jeder einzelnen Stufe darf sich maximal um 10% ändern. [2, Seite 44] [2, Seite 35]

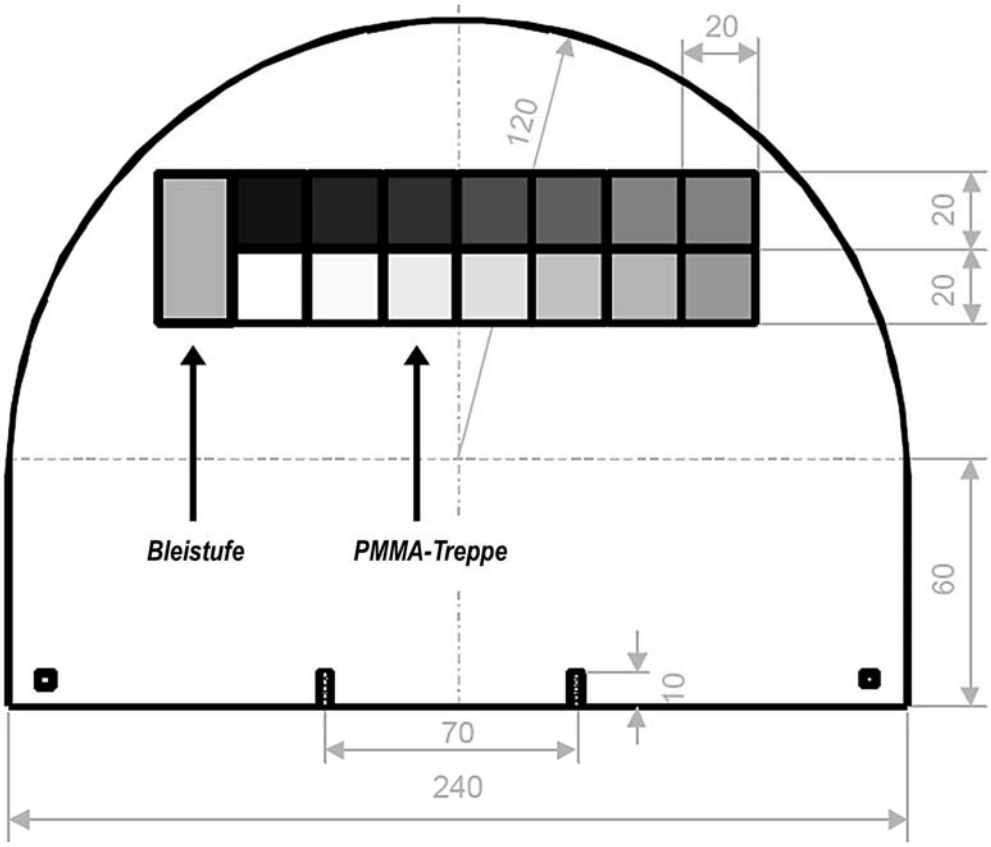

Abbildung 6: Prüfkörper mit Grautreppe und Bleistufe (modifiziert nach [2, Seite 46])

2.4.5 Ausfall von Elementen des Bildempfängersystems

Bei der Abnahmeprüfung und bei Kalibriervorgängen ist eine Defect Pixel Map zu erstellen. Dabei handelt es sich um die Darstellung der aktiven und inaktiven Pixel. Hieraus sind die Anzahl der Defektpixel und deren Anordnung, die als Clustertyp bezeichnet wird, zu entnehmen. Die verschiedenen Clustertypen ergeben sich aus der Beschaffenheit der nächsten und übernächsten Nachbarn des jeweiligen Pixels.

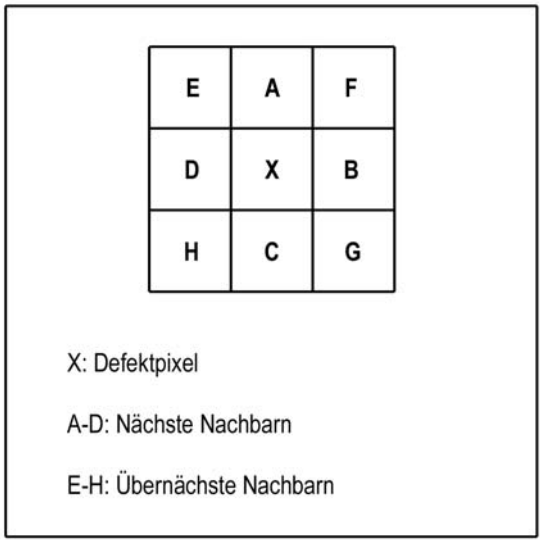

Abbildung 7: Aufteilung der umgebenden Pixel um ein Defektpixel

Man unterscheidet drei Clustertypen:

0-D-Cluster: bestehen aus einem singulären Defektpixel eines Flächendetektors mit aktiven nächsten und übernächsten Nachbarn

1-D-Cluster: bestehen aus einer geraden Anordnung von Defektpixeln eines Flächendetektors in waagerechter und senkrechter Richtung

2-D-Cluster: bestehen aus zusammenhängenden, beliebig angeordneten Defektpixeln eines Flächendetektors

Eine Anordnung, bei der um ein Defektpixel alle nächsten und übernächsten Nachbarn Defektpixel sind, darf nicht auftreten, d.h. dass ein Feld von 3x3 Defektpixeln nicht vorkommen darf. Ist die Anzahl der Defektpixel und deren Clustertypen bestimmt, kann der Unbestimmtheitsindex gemäß Gleichung 2.5 berechnet werden:

$$UI = \frac{\sum_{i=1}^{i=n} (n_{Clustertyp\_i}) * (A_{Pixel}) * (u_{rel}^{i}) * 100}{A_{Detektor}} \tag{2.5}$$

$n_{Clustertyp\_i}$: Anzahl der Pixel eines bestimmten Clustertyps i

A_{Pixel}: aktive Pixelfläche (Pixelapertur) [mm^2]

$A_{Detektor}$: Detektorfläche [mm^2]

u^i_{rel}: relative Unbestimmtheit eines Clustertyps

Der Unbestimmtheitsindex sollte ein Promille nicht überschreiten. Des weiteren dürfen sich in einer Fläche von 1,21mm^2 maximal 20% Defektpixel befinden, unabhängig von ihrer Anordnung. [2, Seite 10,11,38]

3 Umsetzung

3.1 ImageJ und Java

Die Umsetzung der Aufgabenstellung wurde mit Hilfe des Programms ImageJ, Version 1.33u, realisiert. Das Programm wird am U.S. National Institute of Health entwickelt und enthält eine Reihe von Werkzeugen zur Darstellung, Bearbeitung und Analyse von Bildern. ImageJ ist ein kostenfreies im Internet erhältliches Programm, welches sehr einfach durch eigens erstellte Softwarekomponenten, sogenannte Plugins, erweitert werden kann. Neben dem Programm selber finden sich auf der ImageJ-Homepage Updates, Dokumentationen, Testbilder und eine große Anzahl von Plugins. Die Software ist vollständig in Java implementiert, was sie so gut wie plattformunabhängig macht. Es muss auf dem jeweiligen Computer lediglich eine aktuelle Java-Laufzeitumgebung (Java runtime environment "jre") existieren. Die Plugins werden ebenfalls in Java-Code erstellt und über einen eingebundenen Compiler können sie sofort übersetzt und ausgeführt werden. ImageJ unterstützt eine Vielzahl von Dateiformaten, u.a. das DICOM-Bildformat, welches, wie in den Grundlagen beschrieben, in der digitalen medizinischen Bildgebung Standard ist. [7][8, Seite VI, 28, 29]
Die Plugins, welche alle in Java erstellt wurden, und die zu analysierenden Bilder werden grundsätzlich über die ImageJ-Konsole geöffnet. Auf die weiteren Funktion von ImageJ soll hier nicht weiter eingegangen werden, da sie für die im Rahmen dieser Arbeit erstellten Testprogramme keine wesentliche Rolle spielen.

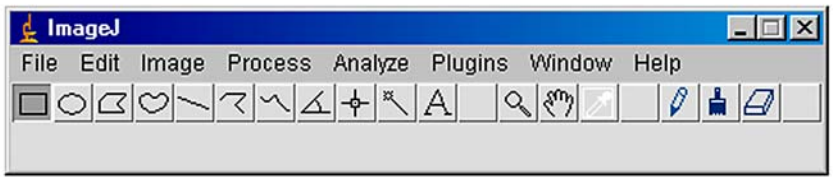

Abbildung 8: ImageJ-Konsole unter Windows 2000

3.2 Aufgaben und Funktion des Programms

3.2.1 Aufgaben des Programms

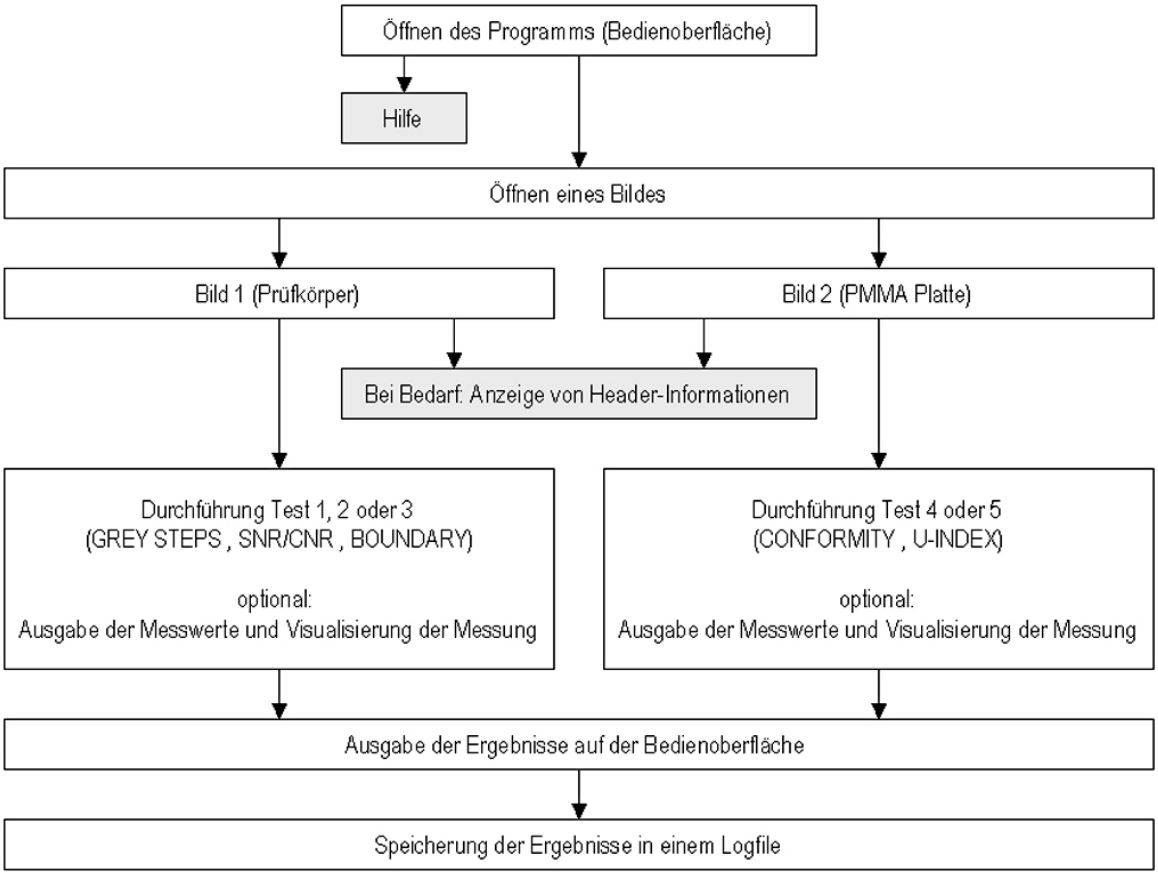

Abbildung 9: Schematische Darstellung des Programmablaufes

Das Programm soll im Wesentlichen dazu dienen, die in Kapitel 2.4 beschriebenen Prüfverfahren zu automatisieren, um es dem Benutzer zu ermöglichen, ohne größere Kenntnisse über die Bildbeschaffenheit, den Aufbau des Prüfkörpers oder die in der PAS beschriebenen Kriterien das Programm zu bedienen und die jeweiligen Tests durchzuführen. Besonderer Wert wird hierbei auf eine automatische und zuverlässige Detektion des Prüfkörpers gelegt. Wichtig ist weiterhin, dass die Ausgabe der Ergebnisse auf ein Minimum an Information reduziert wird und die Ergebnisse direkt auf der Benutzeroberfläche angezeigt werden. Ob ein Test fehlgeschlagen ist oder bestanden wurde, soll in einem Logfile mit Angabe des Datums und der Uhrzeit gespeichert werden können. Für den Fall eines fehlgeschlagenen Tests oder einer nicht erfolgreichen Detektion soll eine Hilfefunktion zur Verfügung stehen, die bei Bedarf aufgerufen werden kann. Die Anzeige der Lage der ROI`s im jeweiligen Bild und die Option, die gemessenen Zahlenwerte in einem Textfile einzusehen, sind zwei weitere Funktionen, die das Programm bieten soll.

3.2.2 Funktion des Programms

Starten des Programms und Programmoberfläche:
Das Hauptprogramm, von welchem aus die einzelnen Tests durchgeführt werden, wird über den Plugin-Ordner in ImageJ gestartet. Die Bedienoberfläche ist in zwei Bereiche aufgeteilt. Im linken, dunkelgrau unterlegten Bereich findet man die Bezeichnungen der einzelnen Tests. Daneben befinden sich die zugehörigen Run-Buttons, neben denen nach Durchführung der Tests der jeweilige Status erscheint. Es gibt drei Statusanzeigen, "OK" für einen bestandenen, "test failed!" für einen nicht bestandenen und "could not run" für einen nicht durchführbaren Test. Über den Button "Save" werden die Ergebnisse der Tests gespeichert.
Im rechten Bereich findet man die Optionen zur Visualisierung und zur Ausgabe der Ergebnisse, sowie eine Hilfe zu jedem Test. Das Hilfefenster besteht zum einen aus einer kurzen Erläuterung und zum anderen aus einem Sample-Image. Mit Hilfe des Info-Buttons kann man sich einige Informationen aus dem DICOM-Header des aktuellen Bildes anzeigen lassen.
Im unteren Bereich ist die Versionsnummer und der Gerätehersteller zu finden, für dessen Bilder die jeweilige Version des Programms ausgelegt ist. Die hier dargestellte Version ist auf Bilder ausgelegt, die mit einem Sectra MDM erstellt wurden.

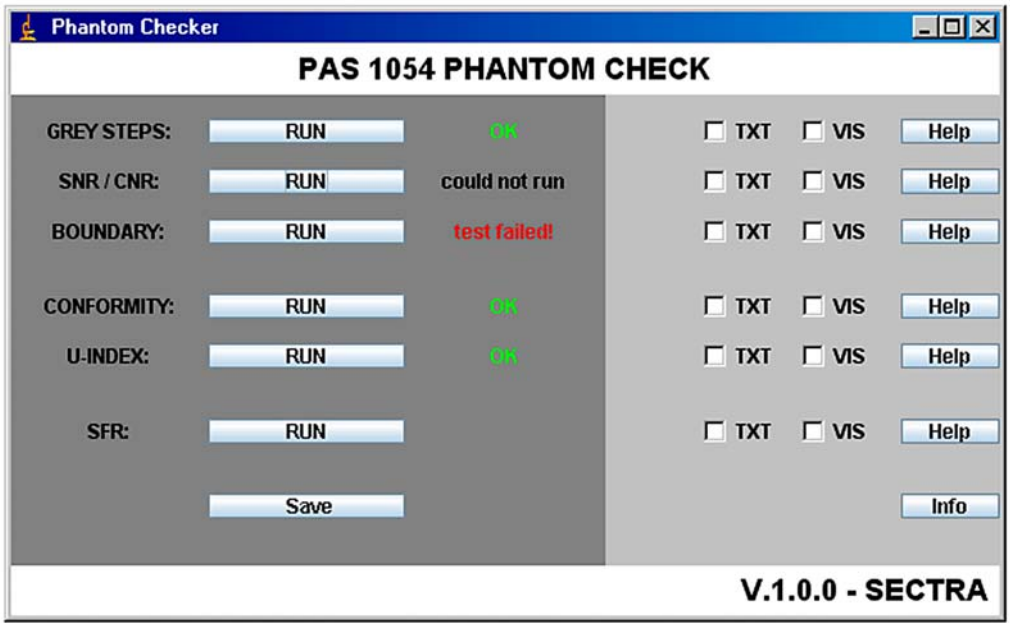

Abbildung 10: Oberfläche des Programms

Durchführung der Prüfungen:

Um einen Test durchzuführen, wird zunächst mit Hilfe der ImageJ-Konsole ein DICOM-Bild geöffnet. Betätigt man einen Run-Button ohne zuvor ein Bild geöffnet zu haben, erscheint die Meldung, dass kein Bild vorhanden ist. Im Nachhinein hat man dann noch die Möglichkeit ein Bild zu öffnen.

Die ersten drei Tests können an ein und derselben Aufnahme durchgeführt werden. In dieser Aufnahme muss der gesamte Grundprüfkörper mit aufgesetzter Strukturplatte abgebildet sein, welcher so platziert ist, dass sich die flache Seite, also die Brustwandseite des Prüfkörpers, am linken oder bevorzugt rechten Bildrand befindet (siehe Abbildung 13, Kapitel 4.3). Es muss der Testeinsatz KRV eingelegt sein.

a) Der Run-Button "GREY STEPS" :

Mit dem Button "GREY STEPS" wird der in Kapitel 2.4.4 beschriebene Test zur Messung des Dynamikumfangs durchgeführt. Der Benutzer wird gebeten, ein Textfile zu öffnen, aus dem die Werte zum Vergleich eingelesen werden. Dieses Textfile kann mit dem Programm *setGreySteps*, welches sich ebenfalls im Plugin-Ordner von ImageJ befindet, erstellt werden. Wählt man die Textausgabe der Ergebnisse, erhält man die gemessenen Grauwerte der 14 Felder, die Grauwerte aus dem zuvor geöffneten Textfile, die Abweichung der Grauwerte voneinander in Prozent, welche maximal 10% betragen darf, das gemessene SRV der einzelnen Felder und das SRV aus dem Textfile.

Results (grey steps)

File Edit

Index	measured grey values	values from txt	aberration in %	measured SNR	SNR from txt
0	2239.44	2199.29	1.82	14.51	14.47
1	2145.34	2110.43	1.58	16.14	16.0
2	2072.63	2039.91	1.48	17.8	17.54
3	2004.89	1977.79	1.23	17.11	16.65
4	1955.69	1928.8	1.22	16.32	15.85
5	1900.85	1877.5	1.06	15.55	15.27
6	2676.85	2602.96	3.35	15.06	14.84
7	2640.2	2566.44	3.35	16.21	16.18
8	2669.14	2592.48	3.48	16.17	16.28
9	2634.3	2563.84	3.2	15.32	15.27
10	2577.27	2512.62	2.93	15.29	15.18
11	2529.19	2468.0	2.78	15.05	14.97
12	2465.88	2410.28	2.52	14.83	14.72
13	2414.95	2364.86	2.27	14.45	14.32

Abbildung 11: Beispiel zur Textausgabe

b) Der Run-Button "SNR/CNR":
Dieser Programmteil führt eine Messung des SNR und des CNR durch, welche im Kapitel 2.4.2 beschrieben sind. Auf einen Programmteil, der einen Vergleich von Werten aus mehreren Bildern durchführt, musste aufgrund von fehlendem Bildmaterial verzichtet werden. Bei einer erfolgreichen Messung erscheint in der Statusanzeige daher lediglich ein "measured". Wählt man die Textausgabe der Messwerte, erhält man den Grauwert die Standardabweichung und das SNR der beiden Felder. Außerdem wird das CNR ermittelt.

c) Der Run-Button "BOUNDARY":
Die Prüfung der thoraxwandseitigen Begrenzung (Kapitel 2.4.1) wird über den Button “BOUNDARY“ durchgeführt. Von den abgebildeten Kugeln müssen hier mindestens fünf in beiden Messbereichen erkannt werden. Als Textausgabe erhält man die Anzahl der jeweils erkannten halben und ganzen Kugeln.

Die Tests “CONFORMITY“ und “U-INDEX´“ können ebenfalls an einer Aufnahme durchgeführt werden. Es wird dazu eine Aufnahme einer 40mm starken, formatfüllenden PMMA-Platte benötigt. Dabei muss sich die Brustwandseite der Aufnahme am rechten Bildrand befinden.

d) Der Run-Button "CONFORMITY":
Mit dem Button “CONFORMITY“ startet man den Test zur Gleichförmigkeit (Kapitel 2.4.3). Als Text-ausgabe erhält man bei diesem Test die Grauwerte der sechs ROI’s, die jeweilige Prozentangabe der Grauwerte in Bezug auf den Mittelwert und die jeweilige Standardabeichung. Keiner der einzelnen Werte darf mehr als 20% vom Mittelwert abweichen.

e) Der Run-Button "U-INDEX":
Die Berechnung des Unbestimmtheitsindexes und die dazugehörigen Prüfungen (Kapitel 2.4.5) werden mit dem Button “U-INDEX“ gestartet. Der berechnete Unbestimmtheitsindex, die gefundenen inaktiven Pixel, die Anzahl der jeweiligen Clustertypen und die Anzahl der 3x3 großen inaktiven Areale stehen abschließend als Textausgabe zur Verfügung.

f) Der Run-Button "SFR":
Dieser Button ist noch nicht belegt. Hier soll später anhand einer Kante die SFR (Spatial Frequency Response) bestimmt werden, welche der MTF (Modulation Transfer Function) entspricht. Dies ist allerdings Thema einer anderen Diplomarbeit

Bei jedem Test kann über die Option der "Visualisierung" überprüft werden, ob an der richtigen Stelle gemessen wurde. Es wird jede ROI des laufenden Prüfverfahrens im aktuellen Bild angezeigt. So kann z.B. bei einem fehlgeschlagenen Test schnell und einfach festgestellt werden, ob die Messorte korrekt sind. Bei der Prüfung des Unbestimmtheitsindexes wird außerdem noch ein Byte-Image angezeigt, in dem die Lage der als nicht aktiv erkannten Pixel zu sehen ist.

4 Aufbau des Programms und Beschreibung des Quellcodes

In diesem Kapitel wird die Funktion des erstellten Quellcodes und die Arbeitsweise des Programms erläutert. Auf wichtige Berechnungen oder Programmbereiche wird hierbei besonders eingegangen. Es empfiehlt sich, die Beschreibung zusammen mit dem Quellcode zu lesen, da dessen Funktion hier Schritt für Schritt beschrieben wird. Im Quellcode selber sind ebenfalls einige Anmerkungen zu den jeweiligen Schritten zu finden. Auf Begriffe aus der Java-Programmierung, die hier verwendet werden, wird nicht weiter eingegangen.

4.1 Interaktion der einzelnen Klassen

Das Programm besteht aus neun Klassen. Die Klasse *phantomChecker_* ist das Hauptprogramm, von welchem aus die einzelnen Tests gestartet werden. Jeder Test wird über eine eigene Klasse durchgeführt, welche den gleichen Namen trägt, wie das zugehörige Prüfverfahren auf der Programmoberfläche. Die Testprogramme wiederum greifen bei Bedarf auf die Klasse *constancyTools* zu, welche Methoden enthält, die häufig verwendet werden, wie z.B. die Bestimmung des mittleren Grauwertes einer ROI. Für die Testprogramme *greySteps*, *snrCnr* und *boundary* muss zunächst eine Detektion des Prüfkörpers vorausgehen, um die Lage des jeweiligen Bereiches, in dem gemessen werden soll, zu berechnen. Hierfür ist die Klasse *detect* zuständig. Diese ist bewusst separat geschrieben worden um Sie, wenn das Programm mit Bildern anderer Geräte laufen soll, austauschen zu können. Mehr dazu ist im Kapitel zur Klasse *detect* zu lesen.

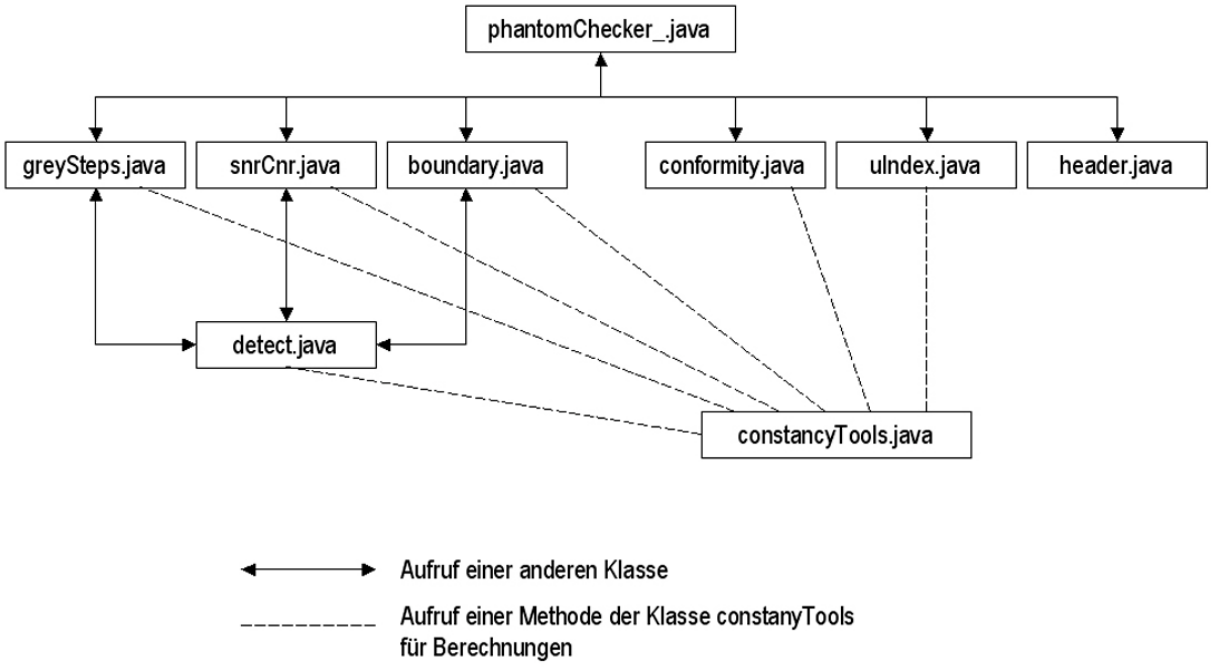

Abbildung 12: Zusammenhang der einzelnen Klassen

4.2 Die Klasse *phantomChecker_*

a) Aufgabe der Klasse *phantomChecker_*

Wie zuvor erwähnt, kann diese Klasse als eine Art Hauptprogramm gesehen werden. Der Unterstrich hinter dem Namen der Klasse sorgt dafür, dass Sie in dem Plugin-Order von ImageJ angezeigt wird. Die anderen Methoden haben diesen Unterstrich nicht, da sie nicht alleine lauffähig sind und von der Klasse *phantomChecker_* aufgerufen werden.

Mit Hilfe der Klasse *phantomChecker_* wird ein Fenster erzeugt, in dem die Bedienoberfläche zu sehen ist, mit der die Funktionen des Programms ausgeführt und gesteuert werden können. Die Hilfe und die Speicherfunktion, welche den angezeigten Status des jeweiligen Tests abruft, sind ebenfalls hier zu finden.

b) Beschreibung des Quellcodes der Klasse *phantomChecker_*

Um die Funktion der einzelnen Buttons und Checkboxen zu ermöglichen, ist in der Klasse der sogenannte *ActionListener* implementiert. Des weiteren enthält Sie die Klasse *PlugInFrame*, um ein Fenster zu erzeugen. Am Anfang werden alle Panels, Buttons, Checkboxen und die Labels für den Status der Tests deklariert. Außerdem werden zwei Variablen vom Typ *boolean* für die Speicherung des Status der Checkboxen deklariert.

Die Klasse hat drei Methoden, den Konstruktor, der den gleichen Namen wie die Klasse trägt, mit dem das Bedienfeld erzeugt wird, die Methode *actionPerformed(...)* um eine Aktion zu verarbeiten und die Methode *save(...)*, die ein Textfile öffnet und die Ergebnisse der Tests darin einträgt.

Im Konstruktor wird zuerst das Frame erstellt, welches im oberen Teil den Titel und im unteren Teil die Versionsnummer enthält. Danach werden zwei Panels erstellt, die später nebeneinander in die Mitte des Frames gelegt werden. Das erste Panel, welches sich auf der linken Seite befindet, ist drei Spalten breit und achtzehn Zeilen hoch. Das zweite Panel ist vier Spalten breit und achtzehn Zeilen hoch.

Die Panels werden Zeile für Zeile mit den jeweiligen Buttons, Labels und Checkboxen gefüllt. Zuerst immer die drei Spalten des linken Panels und danach die vier Spalten des Rechten Panels. Die Namen der jeweiligen Buttons und Checkboxen sind durchnummeriert und im Anfang der Klasse ist dokumentiert, welche Nummer zu welcher Funktion gehört. Durch diesen Aufbau ist es möglich, zu einem späteren Zeitpunkt das Menü beliebig zu erweitern. Dazu muss lediglich zu jedem Panel eine Zeile zugefügt und entsprechend neue Buttons deklariert werden. Gegebenenfalls kann die Größe des Frames neu angepasst werden.

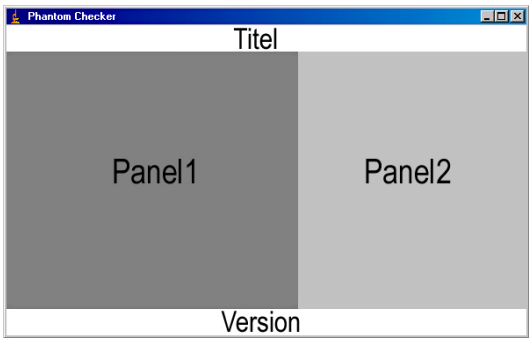

Abbildung 13: Aufteilung innerhalb des Frames

Nach dem Aufbau der Zeilen werden die Panels auf das Frame gelegt und die *ActionListener* der eingebundenen Buttons gestartet. Über den Befehl *show()* wird die Anzeige auf dem Bildschirm ausgelöst.
In der Methode *ActionPerformed(...)* wird die Funktion eines jeden Buttons festgelegt und überprüft, welchen Status die jeweilige Checkbox hat. Zuerst werden die Buttons für die jeweilige Hilfe und die Speicherfunktion überprüft. Wird ein Hilfe-Button betätigt, wird die dazu hinterlegte Information in Form eines JPEG-Bildes geöffnet. Anschließend wird nach einem aktiven Bild gesucht und dieses geöffnet. Ist kein Bild vorhanden, erscheint für die nachfolgenden Buttons eine entsprechende Meldung auf dem Bildschirm und ein Fenster zur Bildauswahl. Dies ist notwendig, da ab hier der Info-Button und die Run-Buttons überprüft werden, für die ein Bild vorhanden sein muss. Bei dem Aufruf eines der Testprogramme wird diesem der Status der Checkboxen übergeben und so mitgeteilt, ob eine Ausgabe der Ergebnisse oder eine Visualisierung erwünscht ist. Der Rückgabewert der einzelnen Programme gibt das Ergebnis des Tests an. Ist der Rückgabewert 1 ist der Test bestanden und das jeweilige Statuslabel wird auf grün und dem entsprechenden Text "OK" gesetzt. Ist der Test fehlgeschlagen wird das Label auf rot und "test failed!" gesetzt. Bei einem Rückgabewert von 3 konnte der Test nicht durchgeführt werden, was durch ein schwarzes "could not run" angezeigt wird.
Die Methode *save(...)*, die über den entsprechenden Button aufgerufen wird, dient zur Speicherung in einem Logfile. Hier wird zuerst ein StringBuffer erzeugt, der am Schluss der Methode alle Zeichen enthält, die angezeigt werden sollen. Danach wird überprüft, ob schon ein Logfile existiert und der Inhalt dessen in den StringBuffer geschrieben. Ist kein Logfile vorhanden, wird dies als Meldung angezeigt. Als nächstes wird das aktuelle Systemdatum und die Systemzeit eingelesen. Diese wird nun mit den Namen der einzelnen Tests und den dazugehörigen Statuslabels hintereinander in den StringBuffer geschrieben. Die Zeilenumbrüche und Tabulatoren werden wie bei Java üblich mit '\n' und '\t' erzeugt. Schließlich folgt eine Linie, die die Einträge optisch voneinander trennen soll. Zum Schluss wird ein Text-Window erzeugt, welches den kompletten Inhalt des StringBuffers anzeigt. Es wird also bei jedem Aufruf ein neues Textfile erstellt, welches die Informationen des alten übernimmt. Beim Speichern wird dann das alte File überschrieben.

4.3 Die Klasse *detect*

a) Aufgabe der Klasse *detect*

Die Klasse *detect* wird benötigt, um den in der PAS beschriebenen Grundprüfkörper im Bild zu erkennen. Sie ist ausschließlich für Bilder gedacht, die mit dem Sectra MDM erstellt wurden. Für eine Detektion in Bildern, die mit Geräten anderer Hersteller erzeugt wurden, ist eine neue Klasse mit einer angepassten Detektion zu erstellen. Messungen an Aufnahmen des Grundprüfkörpers werden in den Klassen *greySteps*, *snrCnr* und *boundary* durchgeführt. Diese starten jeweils die Detektion über den Aufruf von *detect.run(...)*. Wird der Prüfkörper erkannt, ist der Rückgabewert von *detect.run(...)* true, wenn er nicht erkannt wurde, false. Nach der erfolgreichen Detektion können nun die Koordinaten des Mittelpunktes des Testkörpers abgerufen werden, was über die Methoden *detect.getXcenter(...)* und *detect.getYcenter(...)* geschieht. Des weiteren können zwei Rasterwerte abgerufen werden, deren Wert die Anzahl der Pixel angibt, die einer Länge von 5mm im Bild entsprechen. Hierzu stehen die Methoden *detect.getRasterWidth(...)* und *detect.getRasterHeight(...)* zur Verfügung. Mit diesen Angaben kann das rufende Programm nun jeden Testbereich auf dem Grundkörper berechnen. Außerdem besteht die Möglichkeit, über die Methode *detect.getPhantom(...)* die Position des Prüfkörpers als Rectangle ROI abzurufen.

Um eine korrekte Detektion zu gewährleisten, muss der gesamte Prüfkörper im Bild zu sehen sein. Störende Bildinformationen neben dem Prüfkörper müssen vermieden werden. Ein weitere Bedingung ist, dass sich die Thoraxseite des Prüfkörpers auf der rechten oder linken Bildwandseite befindet.

b) Beschreibung des Quellcodes der Klasse *detect*

Zuerst werden die Variablen *phantom* (Rectangle) für den detektierten Bereich, *centerX* (int) und *centerY* (int) für den Mittelpunkt des detektierten Bereiches, rpW (int) und rpH (int) für die Rasterwerte deklariert. Es folgen sieben Methoden, wobei *run(...)* die Startmethode ist. In dieser Startmethode wird zunächst ein Grenzwert berechnet, über den später die Kante des Testkörpers bestimmt wird. Danach wird die Methode *detection(...)* aufgerufen, welche die eigentliche Detektion durchführt. Im Wesentlichen besteht diese Methode aus vier Teilen. Zuerst wird das Bild von links nach rechts gehend Zeile für Zeile nach einem Grauwertsprung über den gegebenen Grenzwert durchsucht (searching for x1). Dies geschieht über zwei ineinander verschachtelte Schleifen. Es werden dabei keine einzelnen Pixelwerte sondern, mit Hilfe der Methode *constancyTools.greyValue()*, ein Mittelwert über eine Fläche von 3x3 Pixeln überprüft. Dies ist nötig, um einen Fehler, z.B. durch Defektpixel, zu vermeiden, was in Kapitel 4.4 genauer beschrieben ist. Der gefundene Punkt ist die linke x-Koordinate des zu detektierenden Bereiches. Auf die gleiche Weise sucht das Programm nun die obere y-Koordinate, die rechte x-Koordinate und die untere y-Koordinate. Mit diesen Werten kann nun das Rechteck *phantom* bestimmt werden.

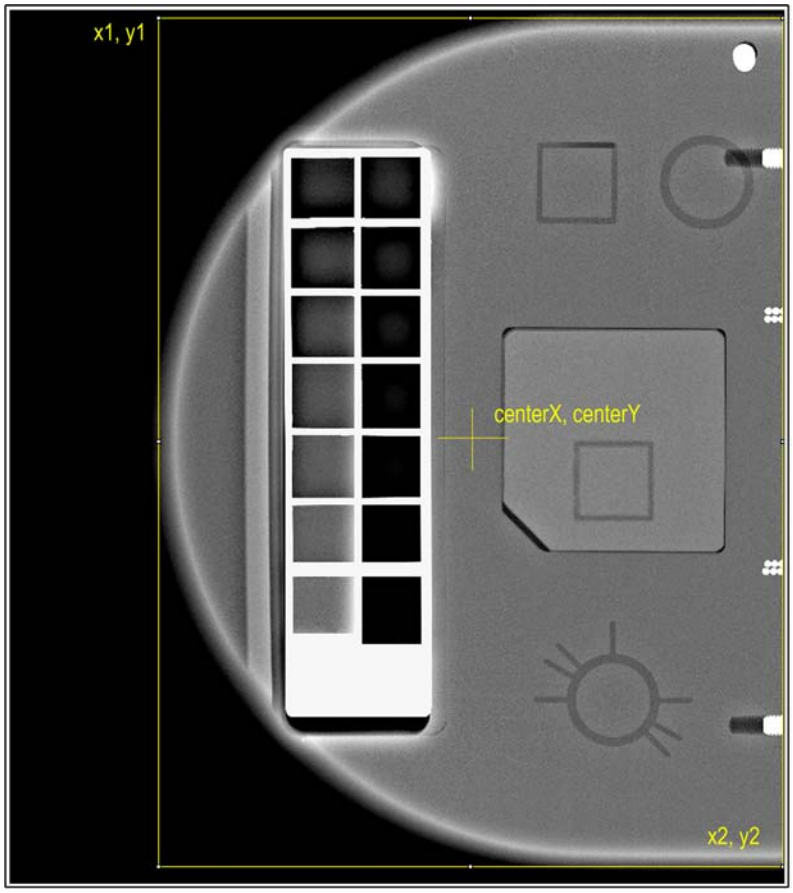

Abbildung 14: Prüfkörperaufnahme mit Rectangle phantom und Koordinaten

Zur Überprüfung, ob es sich bei dem detektierten Bereich um den Prüfkörper handelt, wird der Quotient aus Breite und Höhe des detektierten Bereiches gebildet:

$$Seitenverhältnis = \frac{Breite\_des\_Prüfkörpers}{Höhe\_des\_Prüfkörpers} = \frac{180mm}{240mm} = 0{,}75 \qquad (4.1)$$

Hierbei muss eine gewisse Toleranz gegeben sein, da der Prüfkörper eventuell nicht ganz im Bild ist und der Rand des Prüfkörpers nicht als scharfe Kante abgebildet wird. Daher überprüft das Programm, ob sich der Wert des Seitenverhältnisses zwischen den Werten 0,72 und 0,76 befindet. Diese Werte wurden mit Hilfe von mehreren Aufnahmen verschiedener Prüfkörperdicken ermittelt. Liegt der berechnete Wert außerhalb dieser Grenzwerte, wird das Programm mit dem Rückgabewert false beendet.

Dieser Test soll nur eine vorrübergehende Lösung darstellen, da im Zeitraum dieser Diplomarbeit lediglich Bilder des Prototypen des Prüfkörpers zur Verfügung standen. Eine weitere Möglichkeit wäre die Erkennung des Bereiches für die Offsetmessung, welcher durch den Einsatz einer Bleistufe unterhalb der Grautreppe in der Endversion des Prüfkörpers gegeben ist.

Ist das Seitenverhältnis innerhalb der vorgegebenen Werte, wird überprüft, ob sich der detektierte Bereich in der linken oder der rechten Bildhälfte befindet. Hierzu wird auf beiden Seiten geprüft, wie groß der Abstand, angegeben in Pixeln, zwischen dem Rectangle *phantom* und dem Bildrand ist. Da alle rufenden Programme erwarten, dass sich der Prüfkörper in der rechten Bildhälfte befindet, muss das Bild um 180 Grad gedreht werden, wenn dem nicht so ist. Über einen erneuten Aufruf der Methode *detection(...)* müssen nach einer Drehung die Werte von *phantom* neu gesetzt werden.

Nun werden die Koordinaten des Mittelpunktes und die Rasterwerte geometrisch berechnet. Diese können jeweils über eine entsprechende Methode vom rufenden Programm angefordert werden. Die Namen der Methoden sind in Abschnitt a) aufgeführt. Das Programm endet nach erfolgreicher Detektion mit dem Rückgabewert true.

4.4 Die Klasse *constancyTools*

Diese Klasse enthält Methoden, welche Berechnungen durchführen, die in verschiedenen Tests oder von unterschiedlichen Klassen benötigt werden. Insgesamt gibt es neun Berechnungen:

a) Die Methode *middleGrey(...)*
Die Methode *middleGrey(...)* berechnet den mittleren Grauwert innerhalb einer gegebenen ROI. Der errechnete Wert vom Typ double wird an das rufende Programm zurückgegeben. Die Pixelwerte des Bildes innerhalb der ROI werden einzeln über zwei Schleifen Zeile für Zeile eingelesen und summiert. Anschließend wird dieser Wert durch die Anzahl der eingelesenen Pixelwerte dividiert und ergibt somit den mittleren Grauwert.

b) Die Methode middleGreyModul(...)
Auch in dieser Methode wird der mittlere Grauwert einer ROI berechnet, es werden hierbei allerdings nur die Pixel eingelesen, dessen Grauwerte innerhalb eines bestimmten Bereiches liegen. Der obere und untere Grenzwert dieses Bereiches wird über den mittleren Grauwert einer kleineren ROI ermittelt, die sich im Zentrum der gegebenen ROI befindet. Durch diese Vorgehensweise wird erreicht, dass falls eine ROI nicht direkt auf einem Messfeld liegen sollte und so störende helle oder dunkle Partien in den Randbereichen auftreten, diese nicht in die Messung mit eingehen.
Die Position und die Größe der kleinen ROI wird über die Lage der gegebenen ROI berechnet. Sie soll sich genau in der Mitte dieser befinden und ihre Seitenlänge soll ¼ der Seitenlänge der gegebenen ROI betragen. Es wird der mittlere Grauwert des Inhalts der kleinen ROI über die oben beschriebene Methode berechnet und anhand dessen die beiden Grenzwerte. Diese liegen jeweils 10% über bzw. unter dem Grauwert der kleinen ROI. Innerhalb der gegebenen ROI wird nun die Summe aller Pixelwerte gebildet, die innerhalb der Grenzwerte liegen. Die Variable *counter* zählt hierbei, wie viele Werte summiert werden. Aus diesen beiden Werten kann nun durch Dividieren der mittlere Grauwert ermittelt werden.
Wenn weniger als die Hälfte der Pixel, die sich innerhalb der ROI befinden, aufaddiert wurden, bedeutet dies, dass die Messung zu ungenau ist. Die Variable *modulOK* wird in diesem Fall auf false gesetzt. Über die separate Methode *modulOK(...)* kann jederzeit abgerufen werden, ob die letzte Messung genügend Pixel enthielt.

c) Die Methode *deviation(...)*

Diese Methode berechnet die Standardabweichung der Pixelwerte innerhalb der gegebenen ROI. Die Standardabweichung ist ein Maß für die Streuung der Werte einer Zufallsgröße und charakterisiert so das Rauschen eines digitalen Bildes: [10, Seite 16]

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{4.2}$$

s : Standardabweichung

N : Anzahl der Werte

$\bar{x}$: Mittelwert

x_i : das *i*-te Element in der Menge der Werte

Um die Umsetzung dieser Formel im Programm zu vereinfachen, wurde die Gleichung 4.2 zu Gleichung 4.3 umgeformt:

$$s = \sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N}x_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N}x_i\right)^2\right)} \tag{4.3}$$

Nach der Deklaration der benötigten Variablen wird über zwei Schleifen jeder Pixelwert, der sich innerhalb der ROI befindet, eingelesen und aufsummiert. In derselben Prozedur wird außerdem jeder Pixelwert quadriert und ebenfalls aufsummiert. Des weiteren wird gezählt wie viele Werte eingelesen werden. Nun kann der Mittelwert bestimmt und anschließend die Standardabweichung berechnet und an das rufende Programm zurückgegeben werden.

d) Die Methode deviationModul(...)

Mit dieser Methode wird ebenfalls die Standardabweichung berechnet, allerdings gehen hier wie bei der Methode *middleGreyModul(...)* nur Pixelwerte, die innerhalb zweier Grenzwerte liegen, in die Berechnung mit ein. Die Vorgehensweise ist hier die gleiche wie bei der Methode zur Grauwertmessung. Auch hier wird überprüft, ob genügend Pixel in die Messung mit eingegangen sind und dementsprechend die Variable *modulOK* gesetzt, welche nach einer Messung abgerufen werden kann. Die Methode *modulOK(...)* ist im Quellcode hinter der Methode *deviationModul(...)* zu finden.

e) Die Methoden *max(...)* und *min(...)*

Über zwei Schleifen wird in den Methoden *max(...)* und *min(..)* jeweils der maximale bzw. minimale Pixelwert innerhalb der gegebenen ROI gesucht. Bei Bildern, die mit dem Gerät der Firma Sectra erstellt wurden, entspricht der höchste Pixelwert dem dunkelsten Punkt im Bild.

f) Die Methode *greyValue(...)*

Um die Detektion des Prüfkörpers möglichst fehlerfrei durchzuführen, muss ausgeschlossen werden, dass defekte Pixel fälschlicherweise als Kante des Prüfkörpers interpretiert werden. Zur Vermeidung dessen wird mit der Methode *greyValue(...)* ein mittlerer Grauwert über eine Fläche von 3x3 Pixeln berechnet. Die Methode erhält vom rufendem Programm den Image Processor, eine x-Koordinate und eine y-Koordinate. Die acht Pixel, die um diese Koordinaten herum liegen, sollen mit in die Grauwertberechnung eingehen.

Im Programm wird als erstes überprüft, ob sich das Pixel der gegebenen Koordinaten am Bildrand befindet. Ist dies der Fall, würde das bedeuten, dass einige Nachbarpixel außerhalb des Bildes liegen, also nicht vorhanden sind und somit den Wert Null hätten. Dies würde einen Fehler im mittleren Grauwert erzeugen. Liegt also ein Pixel genau am Bildrand, wird es um ein Pixel in Richtung Bildmitte verschoben, so dass auch hier neun aktive Pixelwerte vorhanden sind. Diese Prüfung findet für alle vier Bildseiten statt. Danach wird jeder Pixelwert über die entsprechende Koordinate eingelesen und aufsummiert, anschließend durch neun dividiert und der sich daraus ergebende Wert zurückgegeben.

g) Die Methode *getSnr(...)*

Das Signal-Rausch-Verhältnis wird durch die Methode *getSnr(...)* berechnet, indem diese einen Wert für die Standardabweichung und den Grauwert erhält. Die Formel, die der Berechnung der Standardabweichung zugrunde liegt, ist in Kapitel 2.4.2 beschrieben.

h) Die Methode *getCnr(...)*

Die Methode *getCnr(...)* erhält zwei Grauwerte, wobei der zweite der höhere ist, und die Standardabweichung. Hieraus wird das Kontrast-Rausch-Verhältnis nach der in Kapitel 2.4.2 beschriebenen Formel berechnet und zurückgegeben.

i) Die Methode middleGreyOval(...)

Zur Berechnung des mittleren Grauwertes einer ovalen ROI wird die Methode middleGreyOval(...) aufgerufen. Da es in der Klasse zur Erstellung einer ovalen ROI keine Methode gibt, mit der die Pixelwerte innerhalb der ovalen ROI abgerufen werden können, erfolgt dies hier mit Hilfe einer Rectangle ROI.

Das Programm erhält vom rufenden Programm also eine rechteckige ROI, berechnet wird jedoch nur der mittlere Grauwert einer Ellipse, welcher sich innerhalb des Rechtecks befindet. Hierzu wird eine ovale ROI erzeugt, welche genau in dem zuvor festgelegten Rechteck liegt. In Java wird eine ovale ROI mit den gleichen Angaben deklariert, wie eine rechteckige ROI. Es werden, wie bei der ersten Grauwert-Methode, alle Pixelwerte innerhalb des Rechtecks eingelesen, aber nur diejenigen aufaddiert, die sich innerhalb der ovalen ROI befinden. Hierzu wird innerhalb der zwei Schleifen geprüft, ob die Koordinaten des gerade eingelesenen Pixels sich innerhalb der ovalen ROI befinden. Die aufaddierten Pixelwerte werden wieder durch die Anzahl der eingelesenen Pixel dividiert und das Ergebnis zurückgegeben.

4.5 Die Klasse *greySteps* und das Plugin *setGreySteps_*

Die Aufgabe dieser Klasse besteht in der Prüfung des Dynamikumfangs. Sie erhält, wie im Kapitel 4.2 beschrieben, vom rufenden Programm zwei Variablen vom Typ boolean, wodurch festgelegt wird, ob eine Visualisierung oder eine Ausgabe der Ergebnisse erwünscht ist. Die Messungen werden an der Grautreppe durchgeführt, welche in den Grundprüfkörper eingesetzt wird. Es erfolgt zunächst die Detektion des Grundprüfkörpers. Wenn diese erfolgreich war, werden die Koordinaten und Rasterwerte abgerufen und hieraus die Position und Größe der Grautreppe berechnet. Mit diesen Werten wird das Rectangle *r* gesetzt und je nach Übergabewert angezeigt. In der Methode *measure(...)* wird nun eine kleine ROI aus den Werten von Rectangle *r* berechnet, welche für die Messungen zuständig ist. Über zwei Schleifen werden anschließend die Messungen durchgeführt. Zuerst wird die rechte Spalte abgearbeitet und die ROI bei jedem Durchgang auf das nächst höhere Feld verschoben. Die Grauwertbestimmung und Standardabweichung wird über die in Kapitel 4.4 beschriebenen Methoden berechnet. Anschließend wird über den Rückgabewert *modulOK(...)* geprüft, ob genügend Pixel gemessen wurden und gegebenenfalls abgebrochen. In der linken Spalte geschieht das gleiche, jedoch mit einer Verschiebung der ROI von oben nach unten. Die Methode endet mit dem Rückgabewert 1 wenn alle Felder korrekt gemessen wurden. Die nächste Schleife berechnet aus den gemessenen Werten das SRV für jedes Feld, danach wird über die Methode *compare(...)* der eigentliche Test durchgeführt.
Über einen Text-Reader wird ein zuvor mit dem separaten Plugin *setGreySteps_* erstelltes Textfile geöffnet, welches vom Benutzer ausgewählt wird. Wird an dieser Stelle nichts geöffnet, bricht das Programm ab und gibt den entsprechenden Wert 3 zurück, da der Test nicht komplett durchgeführt werden konnte.
Das Plugin *setGreySteps_* arbeitet auf die gleiche Weise wie die Klasse *greySteps*, wobei der abschließende Vergleich der gemessenen Werte mit den Werten aus einem Textfile natürlich nicht durchgeführt wird. Die gemessenen Werte werden in dem Textfile so abgespeichert, dass Sie von der Klasse *greySteps* als ImageProcessor geöffnet und spaltenweise eingelesen werden können.
Aus der ersten Spalte werden die Grauwerte und aus der fünften Spalte die Werte für das jeweilige SRV eingelesen. Im nächsten Schritt wird die Differenz des eingelesenen und des gemessenen Wertes in Prozent berechnet. Ist eine Ausgabe erwünscht, wird diese durchgeführt. Am Ende wird überprüft, ob eine höhere Differenz als 10% der verglichenen Werte vorkommt, was, wie im Kapitel 2.4.4 beschrieben, nicht vorkommen darf. Das Ergebnis des Tests wird an die Methode *run(...)* zurückgegeben. Dort wird wieder das Rectangle *r* im Bild angezeigt und das Endergebnis an das Hauptprogramm übergeben.

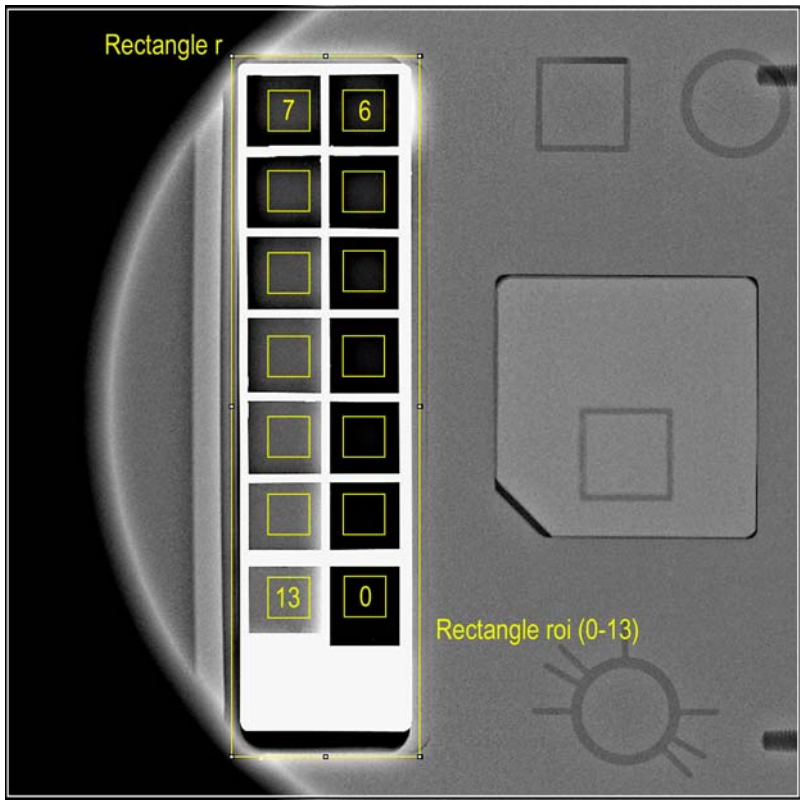

Abbildung 15: Anordnung der ROI's zur Messung auf der Grautreppe

4.6 Die Klasse *snrCnr*

Eine Prüfung, wie sie in Kapitel 2.4.2 beschrieben wird, konnte im Rahmen dieser Diplomarbeit nicht realisiert werden, da nicht genügend Bildmaterial zur Verfügung stand. Es werden in dieser Methode lediglich die Grauwerte, SRV und das KRV in den vorgesehenen Bereichen gemessen.
Genau wie in den zuvor beschriebenen Methoden erhält das Programm die Variablen zur Visualisierung und eine Textoption. Auf eine Beschreibung der Funktion dieser Variablen wird an dieser Stelle verzichtet, da dies bereits zuvor geschehen ist. Jede Ausgabe und Visualisierung wird jedoch im folgenden erwähnt.
Nach der erfolgreichen Detektion des Prüfkörpers wird wieder über die abgerufenen Werte des Detektions-Programms die Position der beiden Messfelder zur Berechnung des CNR und des SNR bestimmt. Das Aluminium-Testfeld befindet sich auf dem entsprechendem Testeinsatz in der Mitte des Prüfkörpers und das PMMA-Testfeld im oberen Bereich. Die Messungen der Grauwerte und der Standardabweichung, die Berechnung der beiden SRV-Werte und anschließend das KRV werden über die entsprechenden Methoden aus der Klasse *constancyTools* durchgeführt. Die Ausgabe der Messwerte erfolgt über ein Resultate-Fenster.
Um den in der PAS beschriebenen Test durchzuführen, müssten diese Messungen an drei Bildern mit unterschiedlicher PMMA-Dicke durchgeführt werden. Die daraus gewonnenen Messwerte werden dann miteinander verglichen (siehe Kapitel 2.4.2). Dies könnte über einen Stack, der die drei Bilder enthält, im Nachhinein realisiert werden.

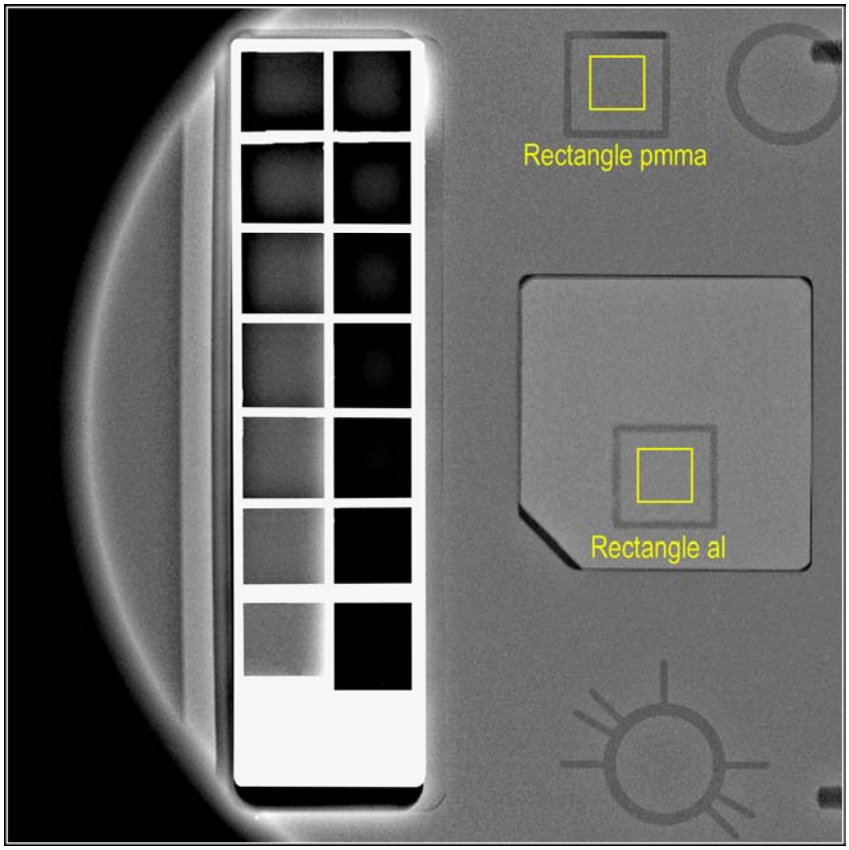

Abbildung 16: Messfelder für die Bestimmung des SRV und KRV

4.7 Die Klasse *boundary*

Mit Hilfe dieser Klasse soll die Anzahl der abgebildeten ganzen und halben Kugeln an der thoraxwandseitigen Begrenzung ermittelt werden (Kapitel 2.4.1).
Zunächst findet wieder die Detektion des Prüfkörpers statt. Mit den gegebenen Werten wird eine ROI berechnet, deren Größe in etwa der Größe einer Kugel entspricht. Außerdem wird die Lage und die Größe der *roi1* berechnet, welche sich am rechten Bildrand befindet. Sie markiert das Areal mit den ersten Kugeln. Nun wird ein neues ImagePlus mit dazugehörigem ByteProcessor in der gleichen Größe der *roi1* erzeugt. Außerdem wird innerhalb der *roi1* der höchste und niedrigste Pixelwert gesucht und davon der Mittelwert berechnet. Dieser wird in den nachfolgenden Schleifen als Grenzwert verwendet. Jeder Pixelwert, der über dem Grenzwert liegt, wird auf 255, was weiß im Bild entspricht, und jeder darunter liegende auf Null gesetzt, was schwarz entspricht. Diese Werte werden in den zuvor erstellten ImageProcessor (*ipArea1*) geschrieben. Somit erhält man ein binäres Bild, in dem die Abbildung der Kugeln den Pixelwert Null haben. Dieses wird angezeigt und anschließend die Methode *ballDetection(...)* zur Detektion der Kugeln gestartet.

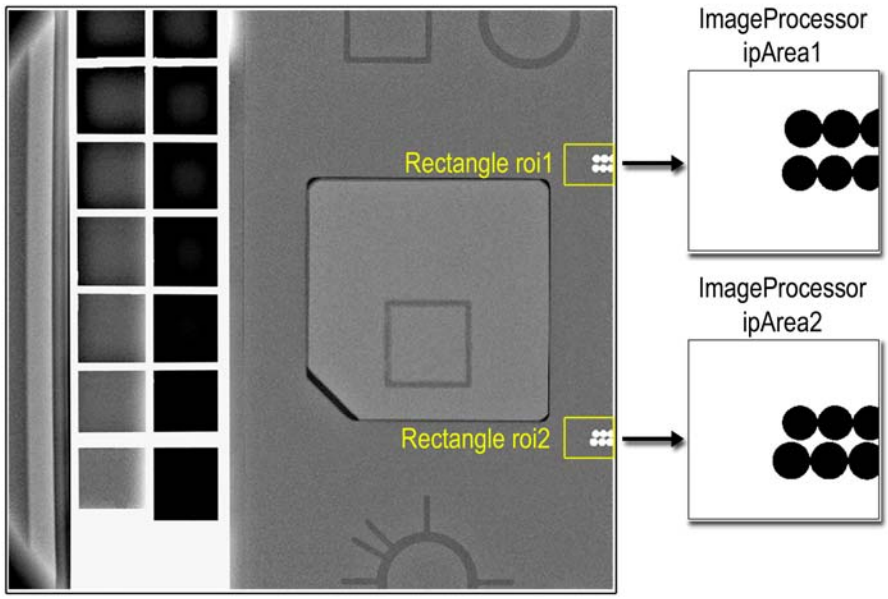

Abbildung 17: ROI's zur Kugeldetektion mit den daraus berechneten binären Bildern

Insgesamt wird in dem neuen Bild nach maximal sechs ganzen und pro Reihe jeweils einer halben Kugel gesucht. Die Suche einer einzelnen Kugel im Bild findet immer über die Methode *getBall(...)* statt. Diese erhält von der rufenden Methode zwei Grenzvariablen für die y-Achse die ihr mitteilen, in welchem Bildbereich sie suchen soll. Bei der ersten Detektion soll im gesamten Bildbereich gesucht werden. Die zu Beginn berechnete ROI, die etwas kleiner als eine Kugel ist, fährt Zeile für Zeile von unten nach oben über das Bild. Sobald die Summe aller Pixelwerte, also der Grauwert des Ovals innerhalb der ROI Null ist, stoppt die Schleife. Die Grauwertberechnung des Ovals wird von der Methode *middleGreyOval(...)* aus der Klasse *constancyTools* durchgeführt. Somit ist die erste Kugel gefunden. Da sich die ROI jedoch am Rand der Kugel befindet, muss diese nun noch in dem gefunden Bereich zentriert werden, was wieder mit Hilfe der Methode *middleGreyOval(...)* geschieht. Das Rectangle des detektierten Bereiches wird an die Methode *ballDetektion(...)* zurückgegeben. Nun hat man durch die erste Kugel auch den Anfang der ersten Reihe von Kugeln gefunden. Daher wird jetzt mit Hilfe der Methode *getballX(...)* ausschließlich auf der x-Achse nach weiteren Kugeln gesucht. Dies geschieht auf die gleiche Weise wie zuvor beschrieben. Ist die letzte Kugel auf der x-Achse gefunden, wird über die Koordinaten der letzten Kugel und den Abstand dieser zum Bildrand berechnet, ob sich eine halbe Kugel am Rand befindet.
Auf die gleiche Weise wie am Anfang der Methode wird nun der Bereich oberhalb der gefundenen Kugelreihe durchsucht, d.h. die Methode *ballDetection(...)* erhält als untere Grenze die obere Kante der bereits detektierten Kugelreihe. Wird hier nichts gefunden, wird unterhalb der ersten Kugelreihe gesucht. Zum Schluss wird die Anzahl der Kugeln in den jeweiligen Reihen addiert, wobei zwei halbe Kugeln eine ganze ergeben. Wenn insgesamt mindestens fünf Kugeln erkannt wurden, gilt der Test für diesen Bereich als bestanden.

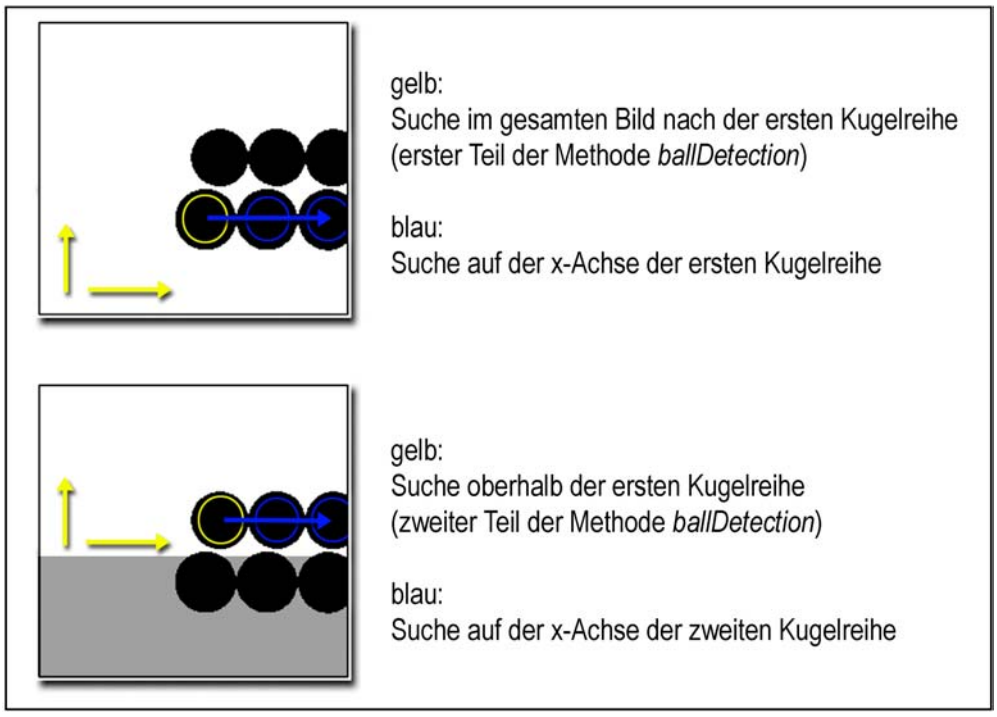

Abbildung 18: Funktion der Kugeldetektion innerhalb der binären Bilder

In der Start Methode wird nun der ImageProcessor *ipArea2* für den zweiten Bereich der Kugelreihen berechnet. Es wird genauso wie zuvor überprüft, ob mindestens fünf Kugeln vorhanden sind. Das Endergebnis, welches angibt, ob in beiden Bereichen genügend Kugeln abgebildet sind, wird an das Hauptprogramm zurückgegeben.

4.8 Die Klasse *uIndex*

Zur Berechnung des Unbestimmtheitsindexes wird die Klasse *uIndex* verwendet. Die Defekt Pixel Map, aus der man die Lage der defekten Pixel entnimmt, wird hier durch ein Byte-Image ersetzt. Dieses Byte-Image stellt die betroffenen Pixel schwarz, was einem Grauwert von Null entspricht, und die fehlerfreien Pixel weiß, also mit einem Grauwert von 255, dar. Zur Erzeugung des Byte-Images wird eine formatfüllende Aufnahme einer PMMA-Platte benötigt. Die hier verwendete Aufnahme wurde nicht mit dem Sectra MDM, sondern mit einem anderen System namens Lorad Selenia angefertigt. Auf dieses Gerät soll allerdings nicht weiter eingegangen werden, da dies für den Aufbau des Programms nicht relevant ist. Am Anfang der Klasse wird auf dieser Aufnahme eine ROI erzeugt, die vom linken oberen und unteren Bildrand jeweils einen Abstand von 10% hat. Dies ist nötig, da hier je nach Aufnahmegerät ein weißer Rand entstehen kann, der nicht in die Messungen mit eingehen darf. Innerhalb dieser ROI wird der mittlere Grauwert berechnet. Alle Pixel, die unterhalb des halben Grauwertes liegen, werden in dem neu erzeugten ImageProcessor schwarz angezeigt. Es handelt sich hierbei nicht um eine wirkliche Defect Pixel Map. Jedoch kann diese Methode als ein brauchbarer Ersatz, der die eigentliche Aufgabe, nämlich die Berechnung des Unbestimmtheitsindexes, ermöglicht, gesehen werden. Die betroffenen Pixel werden daher im Quellcode nicht als "dead Pixel" sondern lediglich als "bad Pixel" beschrieben.

In dem ersten Test fährt eine 16x16 Pixel große ROI den gesamten Bildbereich ab. Dies entspricht in etwa der Fläche von 1,21mm^2, die die PAS vorsieht. Nach jedem Schritt wird in der Methode *testOne(...)* überprüft, ob sich mehr als 50 inaktive, also schwarze, Pixel innerhalb der ROI befinden, was 20% dieser ausmacht. Ist dies nicht der Fall, ist das erste Testkriterium erfüllt.
Im zweiten Test wird der Unbestimmtheitsindex errechnet und gleichzeitig geprüft, ob es Areale mit 3x3 inaktiven Pixeln gibt. Anders als im Kapitel 2.4.5 beschrieben wird hier jedoch zuerst der Unbestimmtheitsindex von jedem einzelnen Pixel berechnet. Dies wird von der Methode *clusterTest(...)* durchgeführt. Sie erhält die Koordinaten des zu prüfenden Pixels. Ein aktiver Nachbar wird in der Methode mit 1 angegeben ein inaktiver mit 0. Zunächst wird überprüft, ob sich das gegebene Pixel am Rand des Bildes befindet und dementsprechend die Nachbarpixel außerhalb des Bildes auf 1 gesetzt. Anschließend wird jeder nächste und übernächste Nachbar überprüft und die aktiven Pixel jeweils aufaddiert (siehe Bild 6 in Kapitel 2.4.5). Der Unbestimmtheitsindex des gegebenen Pixels kann schließlich über folgende Formel berechnet werden:

$$UI_{Pixel} = 1 \Big/ \left(ANN + A\ddot{U}N * \sqrt{2} \right) \tag{4.4}$$

UI_{Pixel}: Unbestimmtheitsindex des gegebenen Pixels
ANN: Aktive nächste Nachbarn
AÜN: Aktive übernächste Nachbarn

Da eine Fläche von 3x3 inaktiven Pixeln laut PAS nicht erlaubt ist, wird in diesem Fall der Unbestimmtheitsindex des Pixels auf 99 gesetzt. Damit wird sicher gestellt, dass der Unbestimmtheitsindex des gesamten Bildes höher ist als ein Promille. Ein Wert von mehr als ein Promille ist das Kriterium dafür, dass dieser Test nicht bestanden ist. Der für jedes Pixel ermittelte Unbestimmtheitsindex wird schließlich an die rufende Methode zurückgegeben.

Über den Unbestimmtheitsindex des einzelnen Pixels kann nun mit Hilfe der unten stehenden Angaben der Clustertyp dieses Pixels ermittelt werden, was im Falle von 1-D- und 2-D-Clustern mit einer gewissen Ungenauigkeit behaftet ist. Diese ist darauf zurückzuführen, dass beim Vorhandensein von zwei defekten nächsten Nachbarn das Pixel, um das diese Nachbarn herum liegen, aufgrund der Lage dieser nächsten Nachbarn sowohl als 1-D- als auch als 2-D-Cluster interpretiert werden kann.

0-D-Cluster:	0	<	UI_{Pixel}	<	0,11
1-D-Cluster:	0,11	<	UI_{Pixel}	<	0,147
2-D-Cluster:	0,147	<	UI_{Pixel}	<	99

Alle ermittelten Unbestimmheitsindexe der einzelnen Pixel werden aufaddiert, durch die gesamte Pixelanzahl des Bildes dividiert und in Promille ausgegeben. Wie bereits erwähnt, muss dieser Wert unter einem Promille liegen, damit der Test als bestanden gilt. Außerdem wird bei Bedarf die Anzahl der einzelnen Clustertypen mit oben beschriebener Ungenauigkeit, die Gesamtzahl der inaktiven Pixel und die Anzahl der Areale mit 3x3 oder mehr inaktiven Pixeln ausgegeben. Die Klasse *uIndex* gibt nun das Ergebnis, welches besagt, ob der erste Test und/oder der zweite Test bestanden ist, an das Hauptprogramm zurück.

4.9 Die Klasse *conformity*

Dieser Programmteil führt den Test zur Gleichförmigkeit durch. Es wird genauso wie bei der Bestimmung des Unbestimmtheitsindexes eine große ROI festgelegt, die den Bereich definiert in dem gemessen werden soll. Innerhalb dieses Bereiches wird nun mit Hilfe einer kleinen ROI überprüft, ob es sich um eine einigermaßen gleichförmige Aufnahme handelt. Die kleine ROI durchläuft hierzu den gegebenen Bereich Pixel für Pixel und prüft, ob der jeweilige Grauwert mehr als die Hälfte vom Mittelwert des gesamten Grauwertes abweicht. Ist dies der Fall, wird das Programm an dieser Stelle mit dem entsprechendem Rückgabewert beendet. Handelt es sich um eine gleichförmige Aufnahme können die Messungen vorgenommen werden.
Ist eine Visualisierung erwünscht, wird nun die große ROI angezeigt. Über die Klassenmethode *getgrey(...)* werden die Messungen durchgeführt und die Felder für die Grauwerte und die Standardabweichung mit Werten gefüllt. Zuerst werden die Startkoordinaten und die Größe der ROI, welche die Messungen durchführt, berechnet. Über jeweils zwei Schleifen werden danach zuerst die drei ROI's in der linken Bildhälfte und danach die drei in der rechten Bildhälfte gemessen.
Durch den Aufruf der methode *getMiddle(...)* wird der Mittelwert berechnet. Nun kann der Test über die Methode *test(...)* durchgeführt werden. Es wird jeder gemessene Wert mit dem Mittelwert verglichen und derjenige, der die höchste Abweichung aufweist, gespeichert. Weicht dieser Wert mehr als 20% vom Mittelwert ab, gibt die Methode ein false, andernfalls ein true zurück.
Es folgt die Methode *display(...)* welche die Messwerte in einem Text Fenster ausgibt, und am Ende der Klasse die Rückgabe des Ergebnisses.

4.10 Die Klasse *header*

Der Header ist bei jeder DICOM-Aufnahme gleich aufgebaut. Am Anfang jeder Zeile findet sich ein Schlüssel, bestehend aus zwei vierstelligen Dezimal-Ziffernfolgen, welche durch ein Komma getrennt sind. Diese charakterisieren den jeweiligen Eintrag. Nach dem Schlüssel kommen zwei Leerzeichen und die Beschreibung des jeweiligen Eintrags. Man weiß also, das die Beschreibung immer an der elften Stelle nach Zeilenanfang beginnt. Danach folgt ein Doppelpunkt. Der eigentliche Eintrag ist somit immer im Bereich vom Doppelpunkt bis zum Zeilenumbruch zu finden.
Um die Informationen aus dem DICOM-Header einzulesen, werden zuerst die entsprechenden Schlüssel der jeweiligen Zeilen in einem String gespeichert. Danach wird der komplette Header des aktuellen Bildes eingelesen. In diesem wird nun nach den gegebenen Schlüsseln gesucht. Ist die Zeile mit dem jeweiligen Schlüssel gefunden, werden die Einträge der Zeile einzeln gespeichert. Zuerst der Schlüssel, dann der Eintrag bis zu dem Doppelpunkt und danach die eigentliche Angabe hinter dem Doppelpunkt bis zum Zeilenende. Dies geschieht für jeden Schlüssel. Nacheinander werden alle gewünschten Informationen in einen Puffer-Speicher geschrieben und anschließend ausgegeben. Möchte man sich weitere Informationen ausgeben lassen, muss das Programm nur mit dem entsprechenden Schlüssel erweitert werden.

5 Ergebnisse

Im Rahmen dieser Diplomarbeit ist es gelungen, ein Programm zu schreiben, welches zu einer erheblichen Vereinfachung von Konstanzprüfungen in der digitalen Mammographie beiträgt. Es hat sich gezeigt, dass die hierfür ausgewählte Software ImageJ und die Programmiersprache Java zur Umsetzung dieser Aufgabenstellung geeignet sind.
Anzumerken ist jedoch, dass es sich bei dem nun vorliegenden Programm um keine endgültige Version, sondern lediglich um einen Prototypen handelt. Ein Grund dafür ist, dass zum Zeitpunkt der Diplomarbeit der endgültige Prüfkörper noch nicht zur Verfügung stand und somit die Detektion bisher auf den Prototypen des Prüfkörpers angepasst ist.
Die Detektion, die in dieser Programmversion realisiert wurde, ist speziell auf Bilder ausgelegt, die mit dem Sectra MDM erstellt wurden. Die übersichtliche Struktur des Programms ermöglicht es jedoch, Erweiterungen problemlos einzufügen oder Anpassungen auf Mammographiesysteme anderer Hersteller durchzuführen. Um die Programme zur Detektion zu schreiben, wurden insgesamt drei verschiedene Aufnahmen des Prüfkörpers verwendet, sowie drei Aufnahmen von PMMA-Platten verschiedener Dicke, die die gleiche Grundfläche aufweisen wie der Grundprüfkörper.
Im Gegensatz zu den oben beschriebenen Aufnahmen wurde die hier verwendete formatfüllende Aufnahme der PMMA-Platte, die in Kapitel 4.8 erwähnt ist, mit dem Gerät Lorad Selenia angefertigt.

Die Übersichtlichkeit der Bedienoberfläche und die Hilfe, welche darin angeboten wird, machen es möglich, das Programm zu nutzen, ohne dass große Vorkenntnisse vorhanden sein müssen. Es wurde besonderer Wert darauf gelegt, die ausgegebenen Informationen und Ergebnisse auf ein Minimum zu reduzieren, um einen möglichst bedienerfreundlichen und zeitsparenden Workflow zu gewährleisten.

6 Schlußfolgerung

Es bieten sich weitere Arbeiten zu diesem Thema an, in denen mit Hilfe des endgültigen Prüfkörpers und unter Verwendung einer größeren Menge von Bilddaten die Genauigkeit des Programms erhöht, bzw. die Funktionen erweitert werden können. Eine mögliche Erweiterung wäre z.B. das Ersetzen des Logfiles durch eine geeignete Datenbank. An der Erstellung des Programmteils zur Berechnung der SFR wird derzeit gearbeitet. Dieser Programmteil wird so gestaltet, dass er in das bestehende Programm eingebunden werden kann.

7 Anhang

7.1 Abkürzungen

DICOM	Digital Imaging and Communications in Medicine
HIS	Hospital Information System
KRV	Kontrast-Rausch-Verhältnis, en: Contrast/Signal difference to Noise Ratio, CNR
MDM	Micro Dose Mammography
MTF	Modulation Transfer Function
OSI	Open System Interconnection
PAS	Publicly Available Specification
PMMA	Polymethylmetacrylat
RIS	Radiology Information System
ROI	Region of Interest
SFR	Spatial Frequency Response
SRV	Signal-Rausch-Verhältnis, en: Signal Noise Ratio, SNR

7.2 Literaturverzeichnis

[1] Strahlenschutzkommission:
Digitale Mammographie in der kurativen Anwendung und im Screening
verabschiedet in der 197. Sitzung der Strahlenschutzkommission am 16./17. Dezember 2004

[2] PAS 1054:
Anforderungen und Prüfverfahren für digitale Mammographie-Einrichtungen
Beuth Verlag GmbH, Berlin 2005

[3] Jürgen Braun:
Bildgebende Verfahren in der Medizin, Teil 4:Mammographie Bildqualität von analogen und digitalen Daten

[4] Bernhardt, T.M., Ludwig, K.:
Digitale Flachdetektorsysteme
Radiologie up2date 1 (2002) 45-57

[5] http://www.teleradiologie.de
(August 2005)

[6] Normenausschuss Radiologie NAR:
Abnahmeprüfungen Mammographie
(2002)

[7] http://rsb.info.nih.gov//j/
(August 2005)

[8] Burger, W., Burge, M.J.:
Digitale Bildverarbeitung.
Springer-Verlag, Berlin, Heidelberg 2005

[9] Nührmann, D., Meyer-Veith, T.:
Mathematik für Elektroniker
Franzis GmbH Verlag, München 1989

[10] DIN V ENV 13005:
Leitfaden zur Angabe der Unsicherheit beim Messen
Beuth Verlag GmbH, Berlin 1999

7.3 Abbildungsverzeichnis

7.4 Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Köln, den 19.08.2005

Andreas Schreiber

7.5 Sperrvermerk

Die vorgelegte Arbeit unterliegt keinem Sperrvermerk.

7.6 Weitergabeerklärung

Ich erkläre hiermit mein Einverständnis, dass das vorliegende Exemplar meiner Abschlussarbeit oder eine Kopie hiervon für wissenschaftliche Zwecke verwendet werden darf.

Köln, den 19.08.2005

Andreas Schreiber

7.7 Inhaltsverzeichnis der CD

- ImageJ_ver1.33u_jdk1.5_win
- installation_help
- phantomChecker
- phantomChecker_help_images
- sample_images_DICOM
- thesis